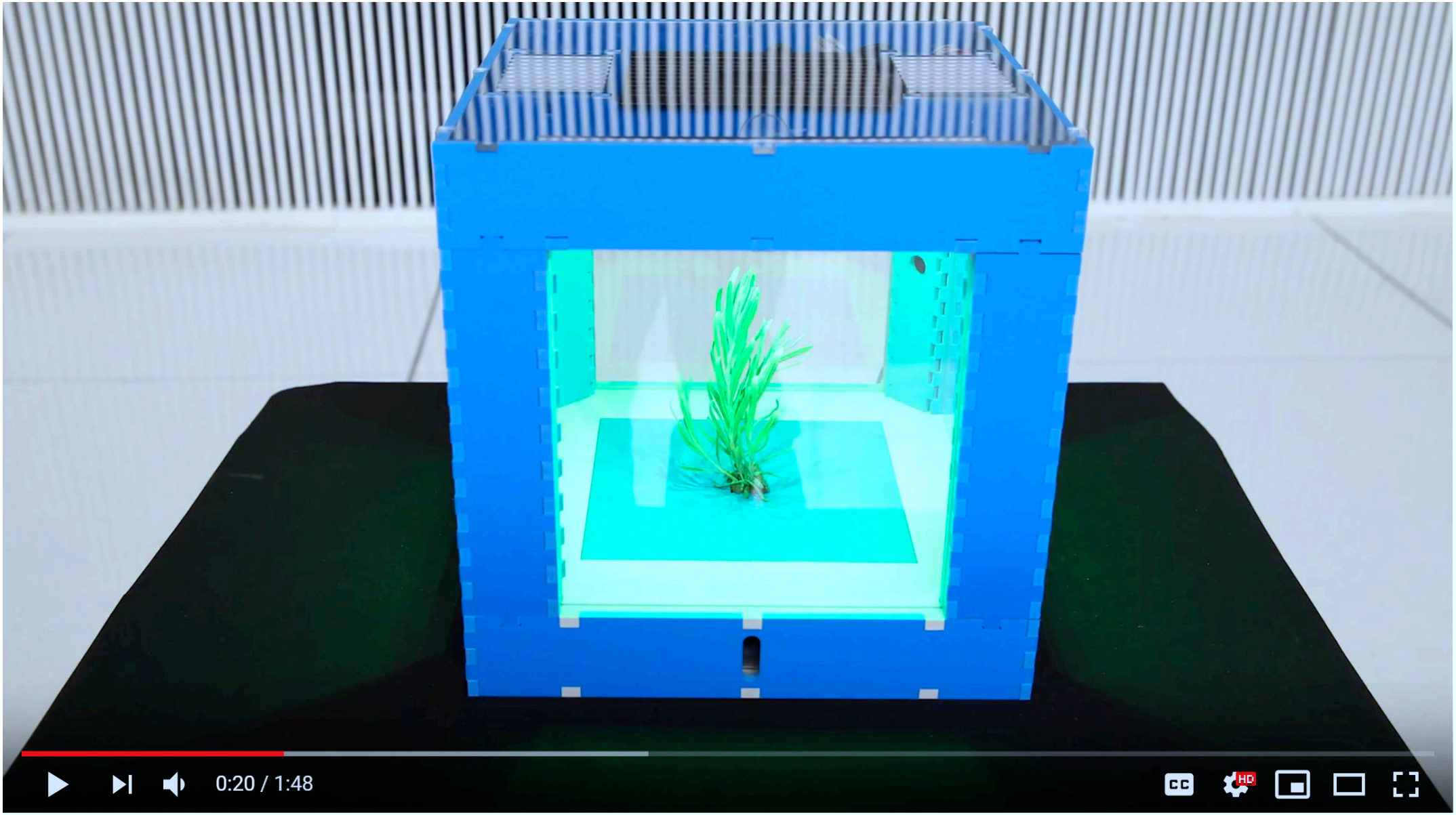
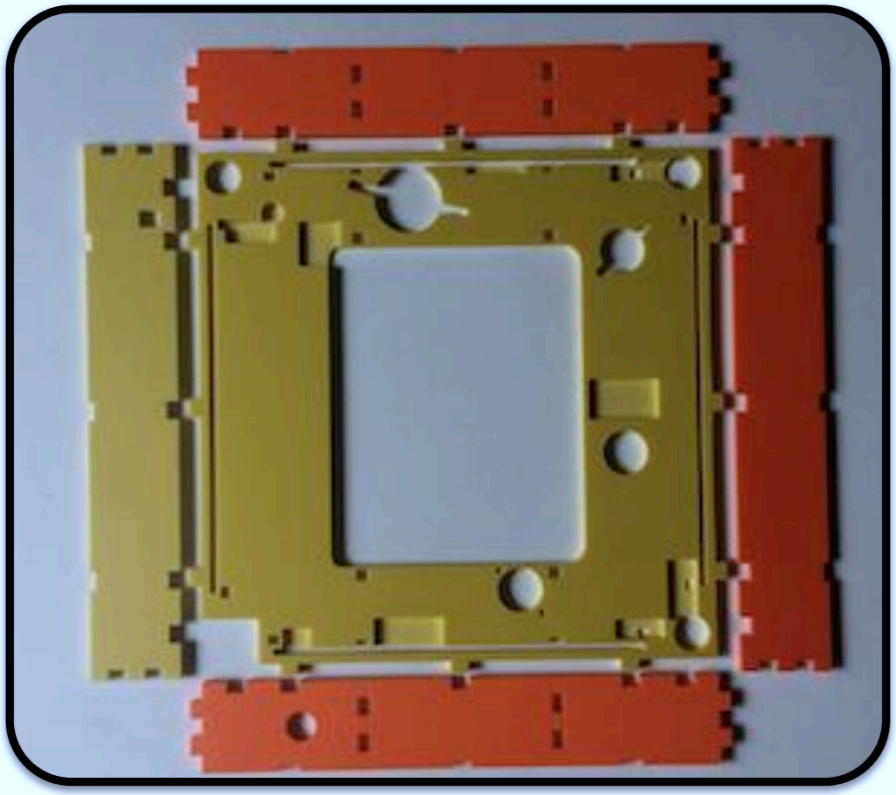
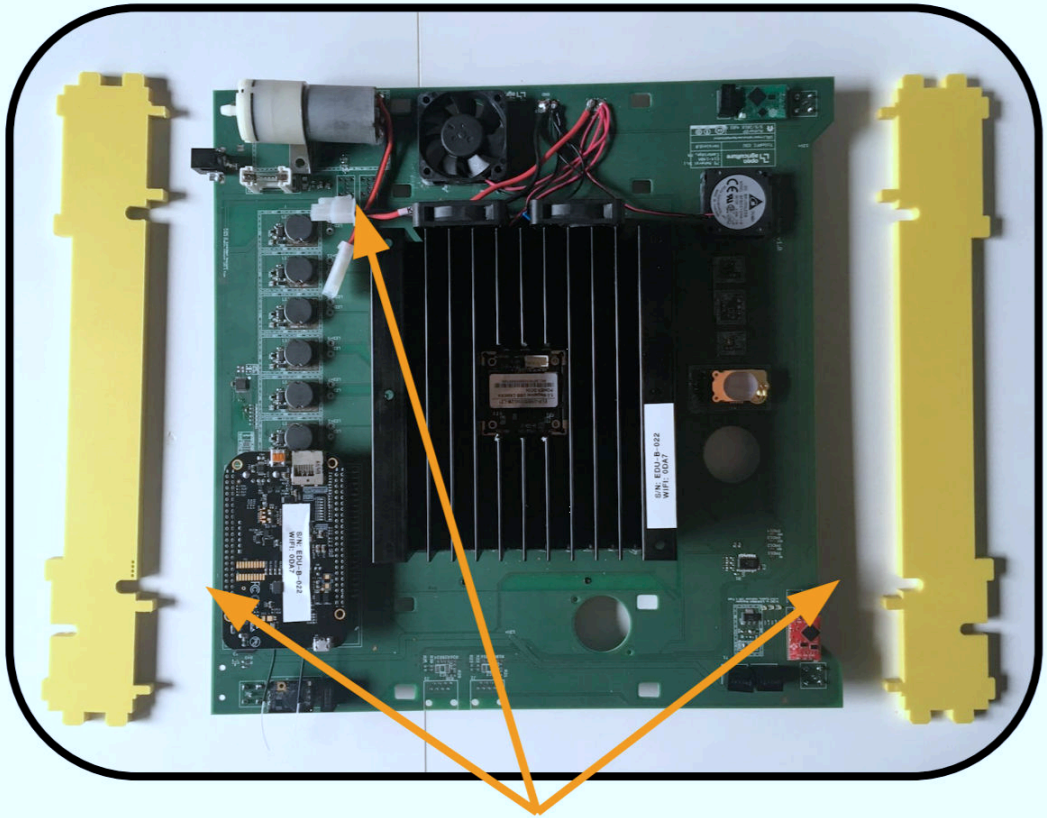


Personal Food Computer v3.0 - the "PFC_EDU"

1. Set out the underside of the top assembly and its side pieces as shown. Each of the cutouts is designed to fit certain components from the circuit board (the Central Nervous System, or CNS).



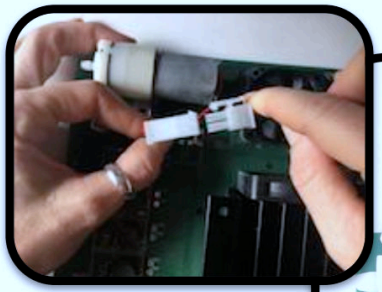


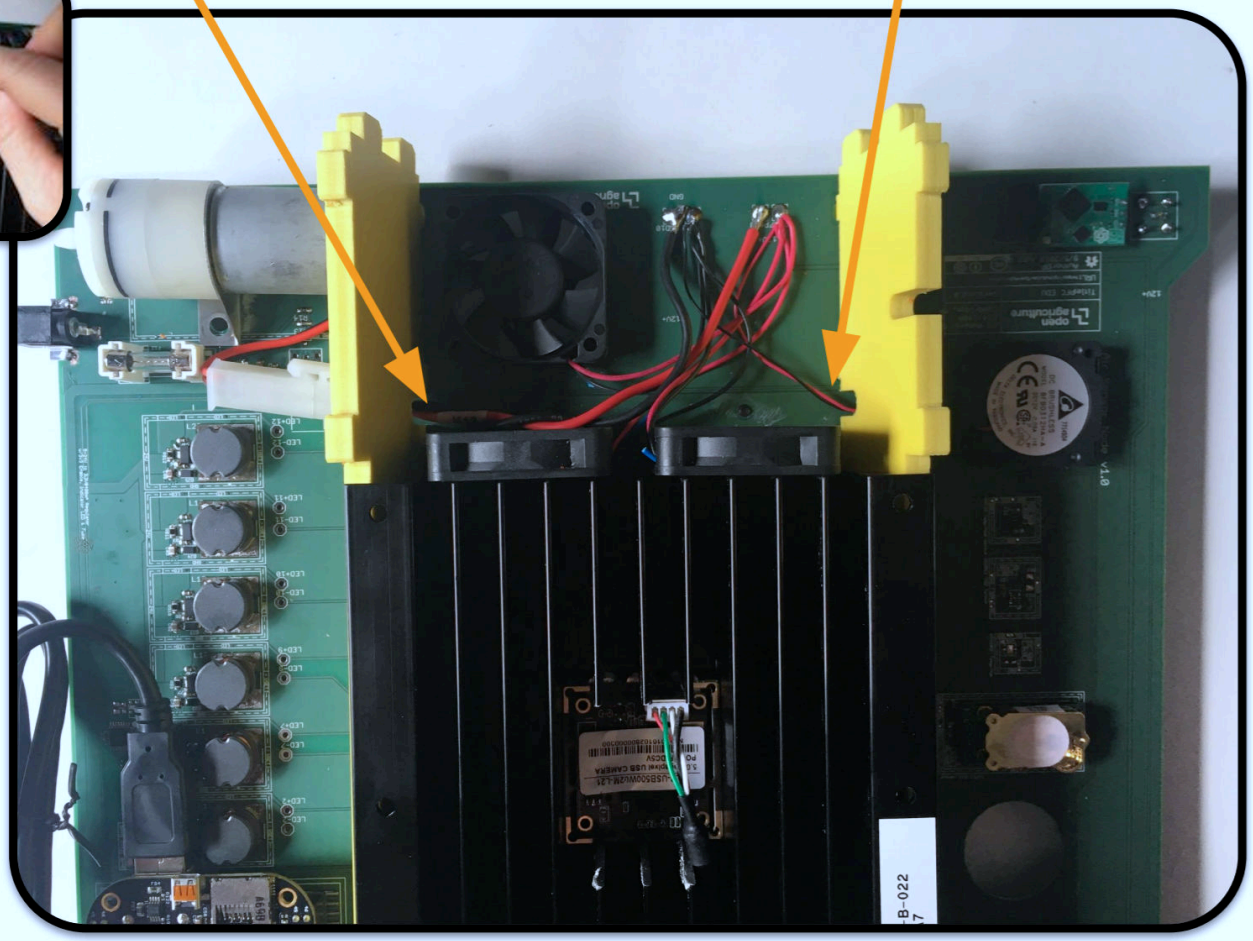
2. Set the CNS board on top, carefully aligning the circuit board components with their cutouts and grooves.

3. Set out the two inner supports, as shown.
4. For this build guide, we'll continue to keep the air pump and power adapter at the top left, for reference

1. Connect/plug in the air pump wires* before setting the 4 skinny tabs on the bottom of the inner supports into their corresponding slots on the circuit board. You will need to angle them in. *Note: Some PFCs may have support pieces already in place. Lucky ducks!*
2. Thread the red and black air sensor wires (top right) and air pump wires (top left*) through the small, tunnel-shaped cut-outs - or the “dog house” - on the bottom of each the supports.

*

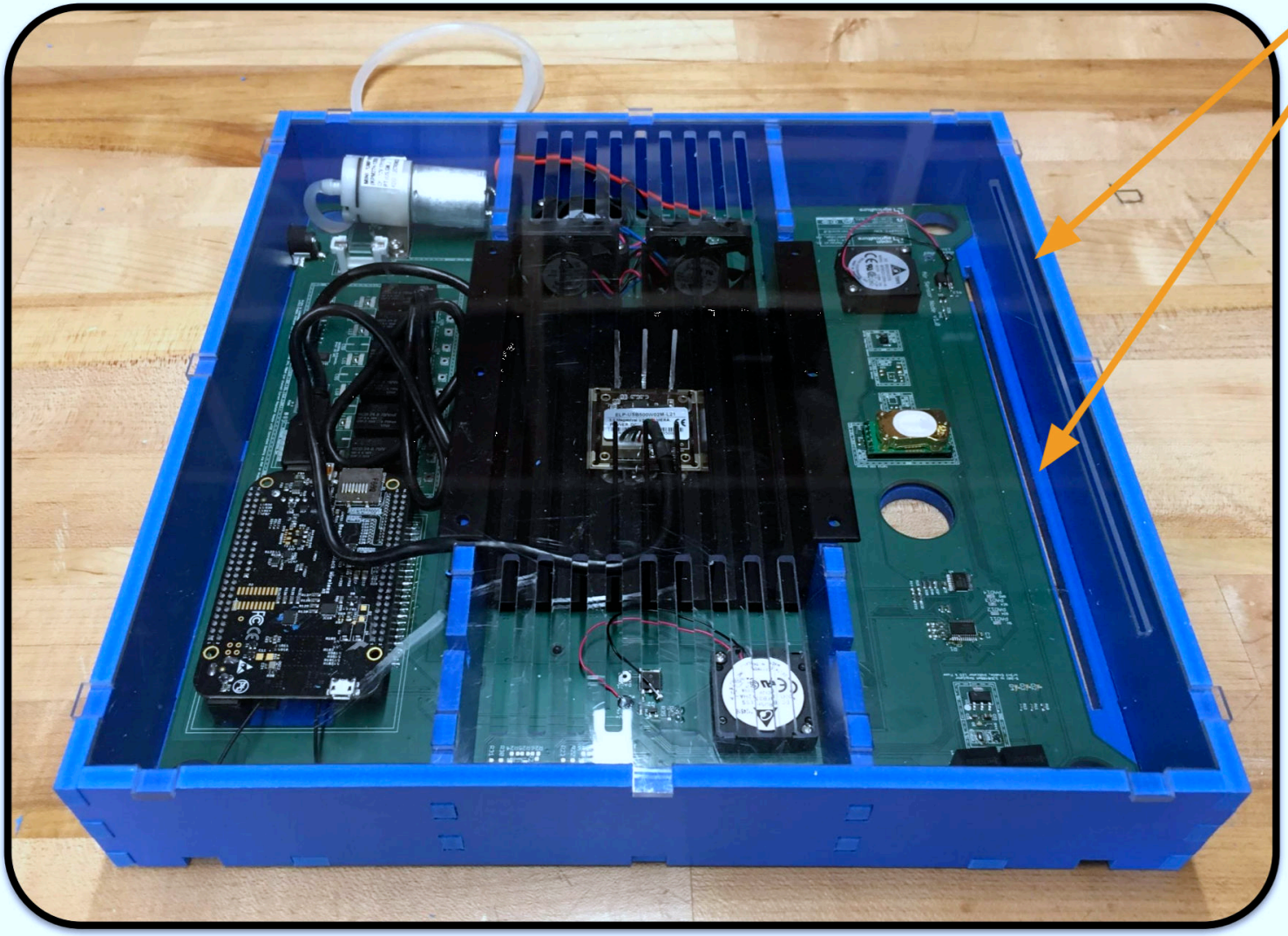




LEARNING
iNVENT & iNSPIRE

open agriculture

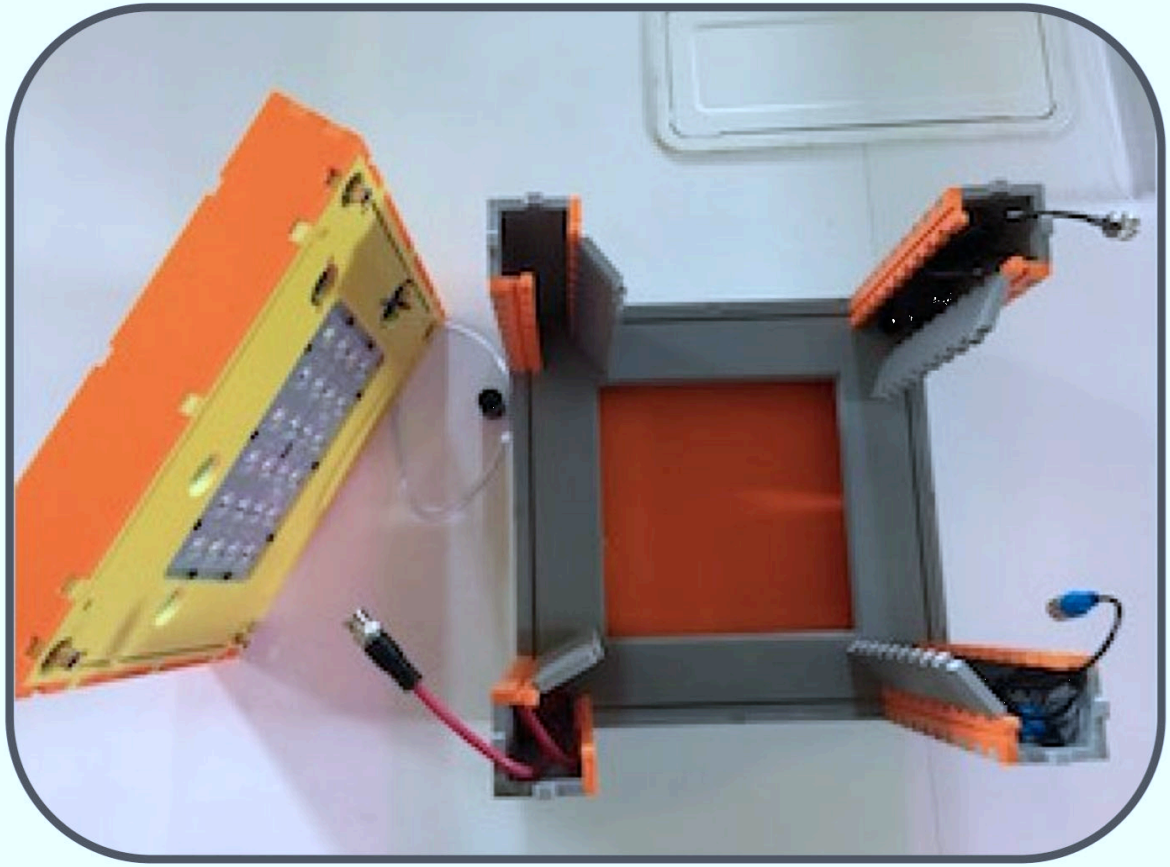
close up the top assembly. Make sure that the slot in the clear plastic cover aligns with the slot in the circuit board base.



2. Set the top assembly aside.

1. Thread the pH sensor cable *<u>up</u> through bottom right column* (blue cable cap, bottom right of CNS) and leave about 3” coming out of the top.
2. Thread the water thermometer cable up through bottom left column (red cable, bottom left of CNS) and leave about 3” coming out of the top.
3. Your air stone tubing is already sticking out of the top left of the top assembly. Thread the tube down through the top left column, and down into the reservoir.

Note: Like the air stone, the ends of the red water temp probe, the EC probe, and the pH probe should be threaded to eventually sit all the way inside the reservoir. The EC and pH probes have caps on them that prevent them from fitting inside the bottom assembly holes, and will need to be removed when you fill the reservoir for the first time, get growing.



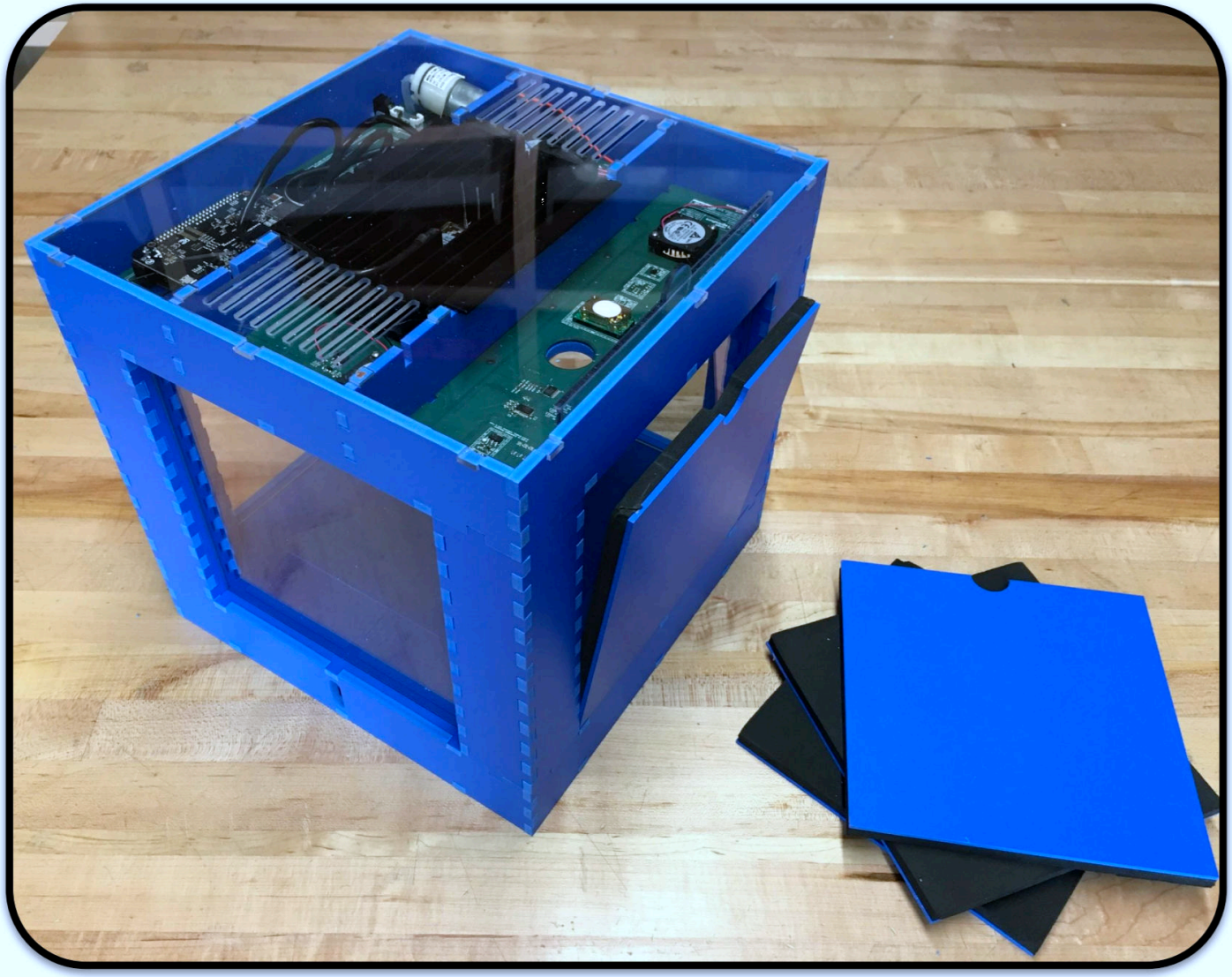
1. Gently lift and rotate the top assembly and place it on top of the columns. Make sure that the air pump is in the top left, and that the slot in the top is on the right.
2. Attach the black EC sensor cable (coming through the top right column) to the CNS.
3. Attach the blue pH probe (coming through the bottom right column) to the CNS.
4. Attach the red Water Thermometer (coming through the bottom left column) to the CNS.

Final Assembly: Step 5

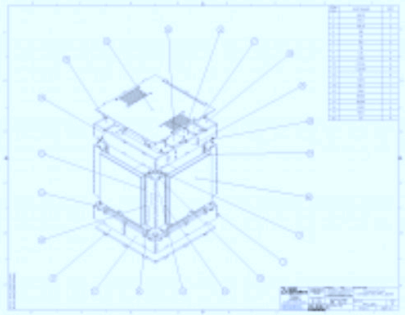
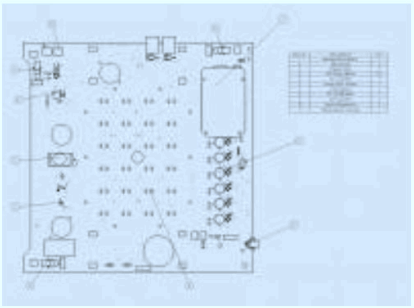
1. Congratulations!! Sit back and admire your work.

2. The foam-padded sides will be used once you're growing, to help keep the climate at the desired temperature, and to help keep the light levels constant.

3. When you're ready to use them, simply push them into place.



PFC_EDU Bill of Materials and Design


The files you'll need to build a PFC_EDU from scratch are below:

- PFC 3.0 (PFC_EDU) Specifications Sheet
- PFC_EDU Assembly Bill of Materials (total unit BOM, Google spreadsheet)

A baseline PFC 3.0 can be assembled for about $500 with the raw materials below. For folks who want to outsource more advanced components to professional manufacturers, see "Getting parts of a PFC_EDU manufactured" below.

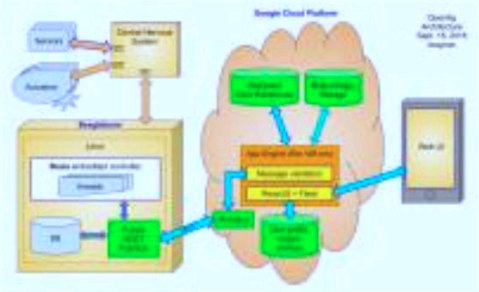
1. Main chassis (frame cut by CNC)
2. Water reservoir (laser cut or vacuum-formed)
3. Air stone and hose
4. A completed "Central Nervous System" (CNS), which includes a printed circuit board (PCB) and
   - BeagleBone Black
   - LED lights
   - Fans
   - Temperature, humidity and C02 sensors
   - Air pump
   - USB camera
   - Power supply and cord
   - For more advanced environmental data, the PFC_EDU CNS can be modified to include these optional sensors:
     - EC probe (a conductivity sensor for nutrient concentration)
     - pH probe
     - Water temperature probe

Software

All the details for getting your BeagleBone Black ready, software installed, configured and updated are available in the ReadMe's at the links below.

- OpenAg Brain (Django/Python)
- OpenAg Cloud (Python/React/Flask)

OpenAg Cloud Architecture



Architecture description

Hardware

Electrical

Files for circuit board, sensors, and actuators (“central nervous system”)

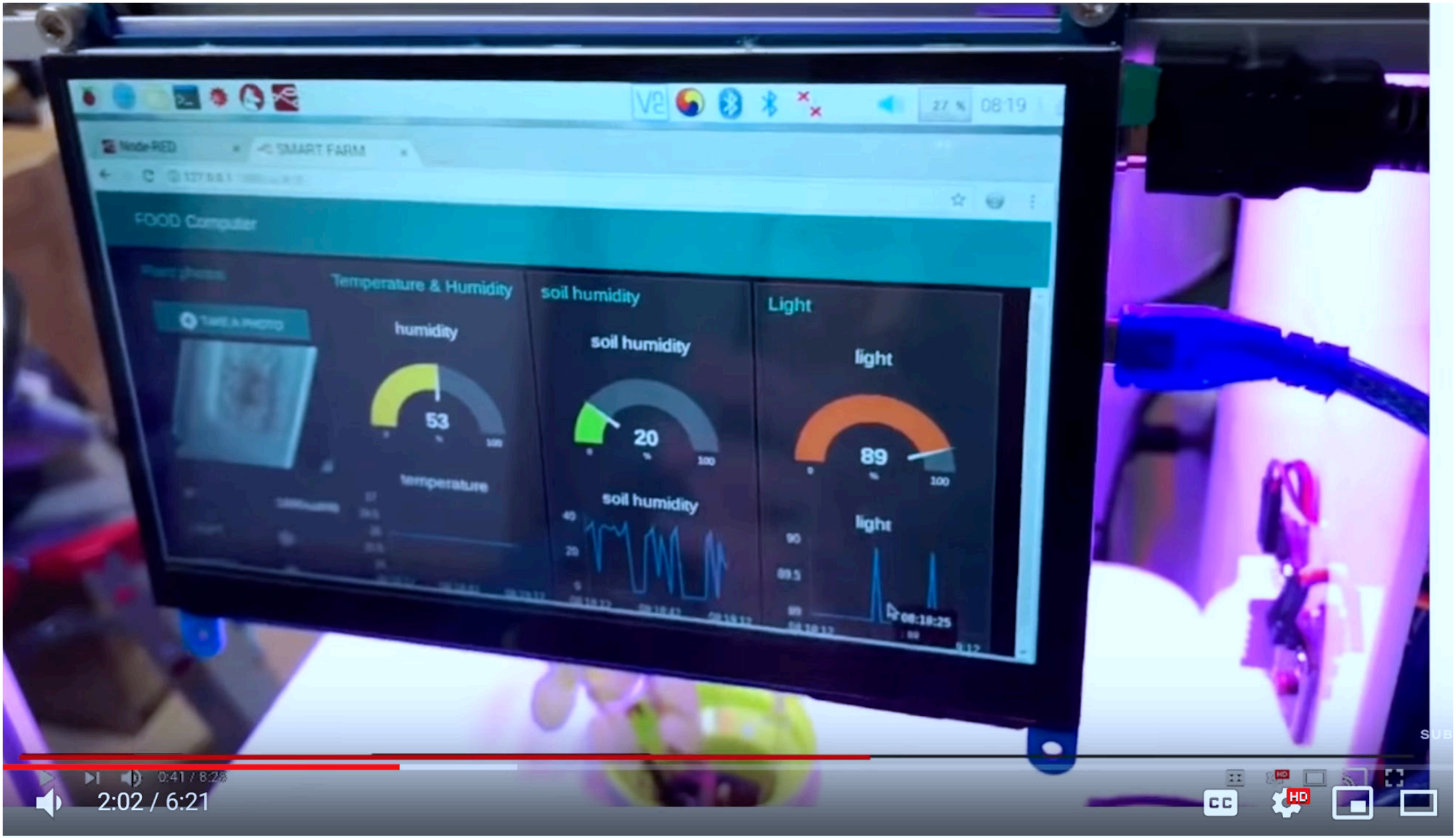
- Circuit Board Design files & Assembly Bill of Materials
- Optional Sensors Bill of Materials* (Google Spreadsheet)

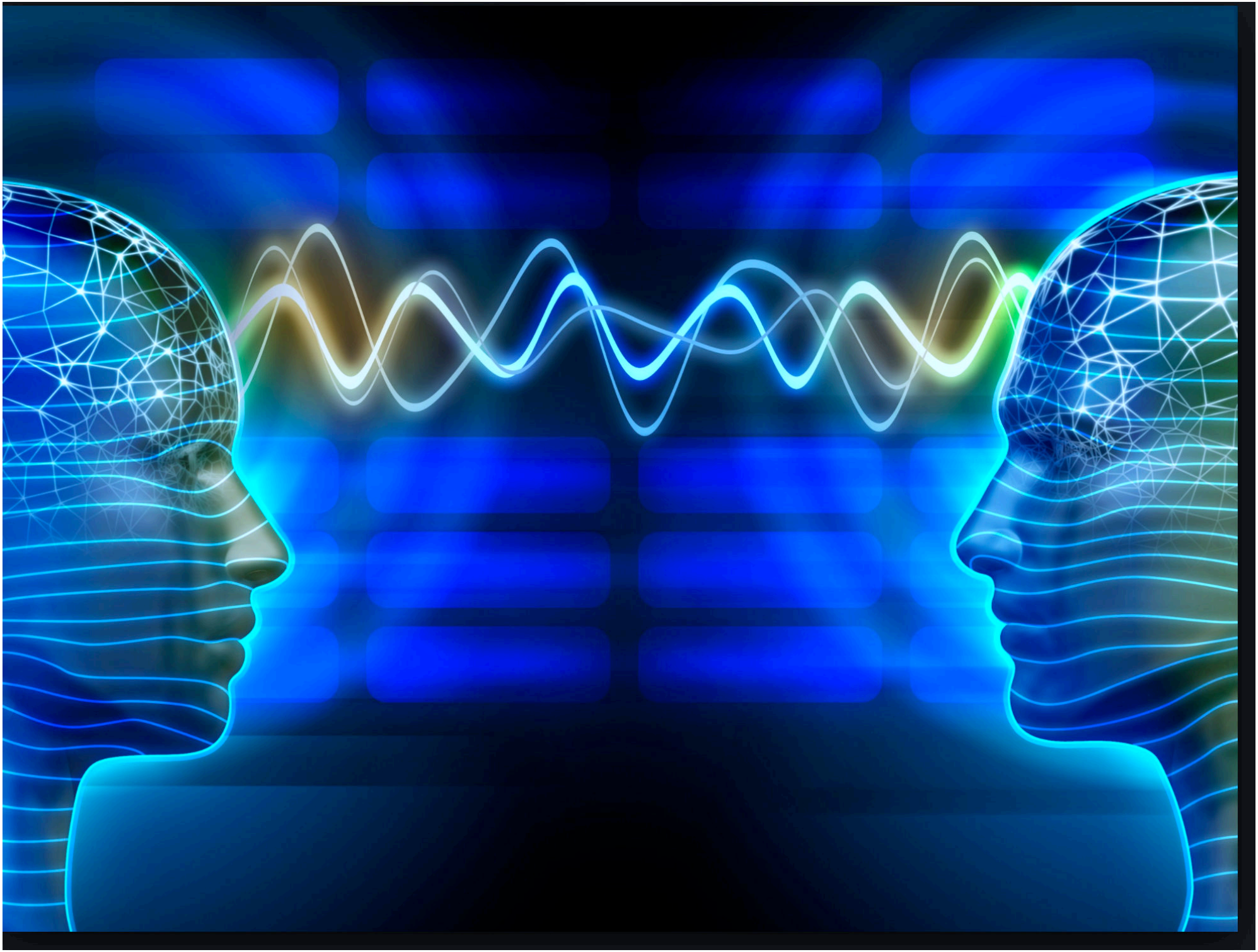
**note: CO2 and environmental temperature/humidity sensors are included in the baseline CNS BOM. The pH, EC, water temperature and camera are optional and included in the Optional Sensors BOM.*

Mechanical/Frame


Seeker
SUBSCRIBE
4:04 / 6:21


Seeker
SUBSCRIBE
5:00 / 6:21


Node-RED
SMART FARM
FOOD Computer
Temperature & Humidity
humidity
53
temperature
soil humidity
soil humidity
20
soil humidity
Light
light
89
light
0:41 / 8:28
2:02 / 6:21



1000 × 562 - Images may be subject to copyright. Learn More

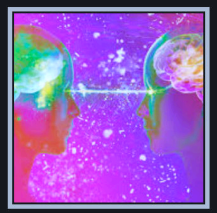
Twin Flame telepathy and all sorts of ...

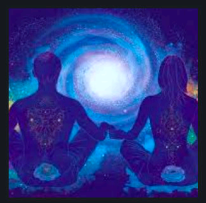
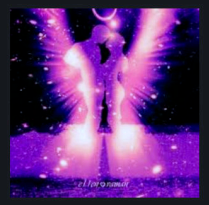
Tribe & Twin Flame Program

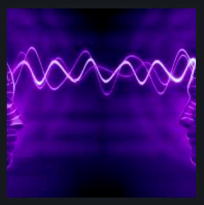
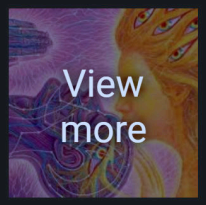
If you are new to this website I highly recommend that you download the free Gangsta Goddesses Manifesto. Reading it will give you a better understanding of ...

Visit | Add to | Collections | Share

Related images:



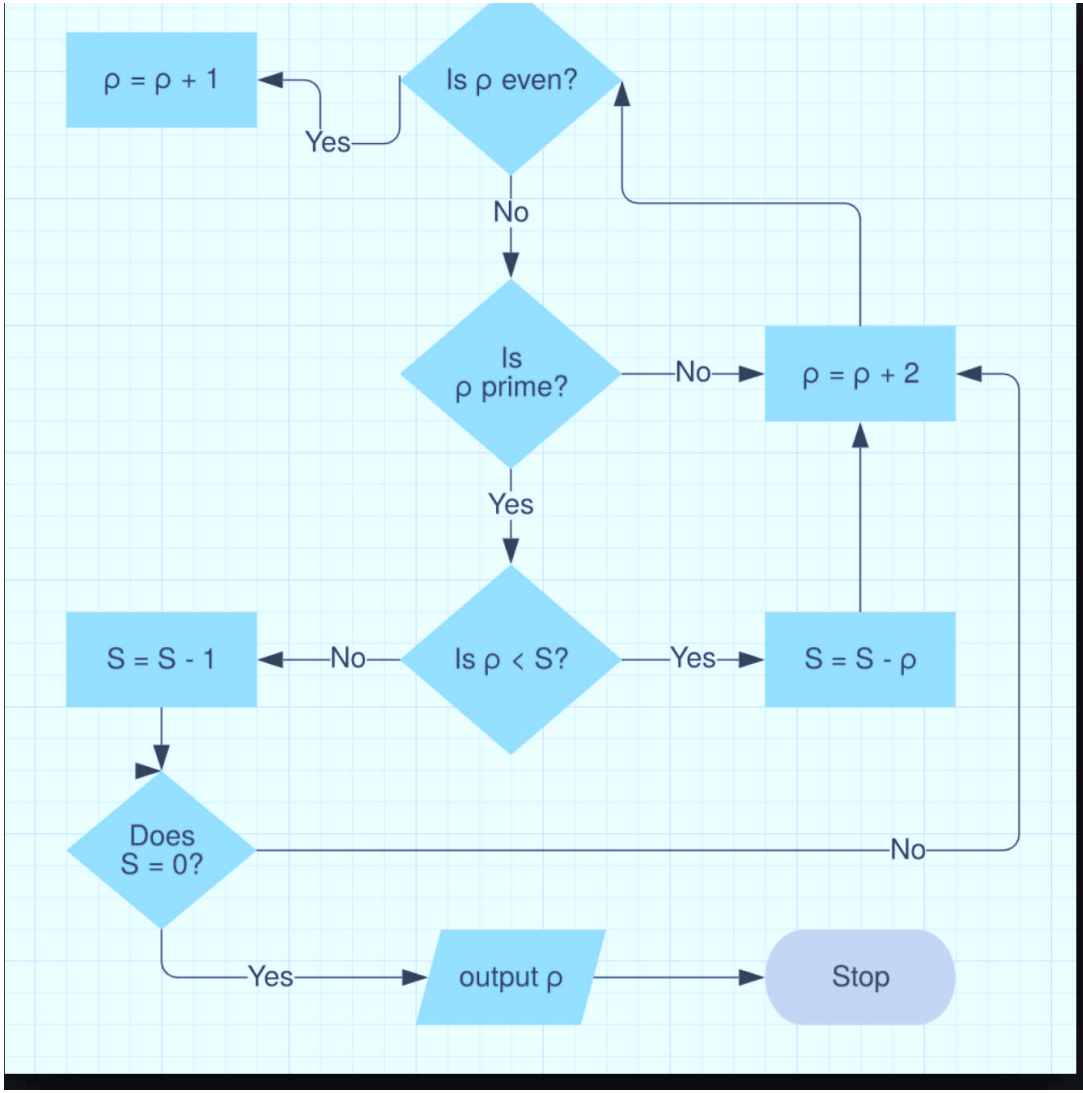
Reality

Reality is a consciousness program (hologram, simulation, illusion, dream) created by digital codes. Numbers, numeric codes, define our existence and experiences. Human DNA, our genetic memory, triggers (remembers) by digital codes at specific times and frequencies as we experience. Those codes awaken the mind to the change and evolution of consciousness.

The brain is an electrochemical machine (computer)
that processes through binary code zeroes and ones
that create patterns of experiences and realities.

The illusion of physical reality is created by the patterns of the
Fibonacci Sequence - the Golden Spiral of Consciousness
consisting of zeros and ones that align with the brain.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

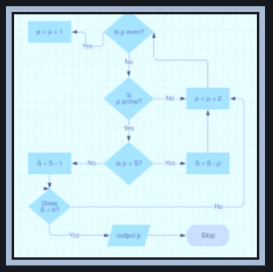
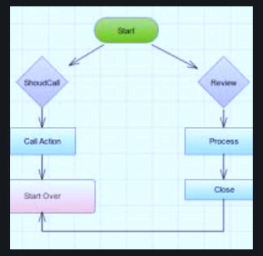
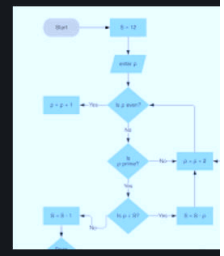


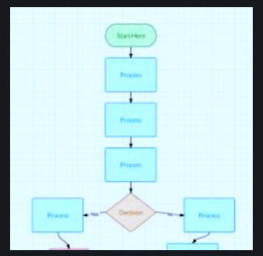
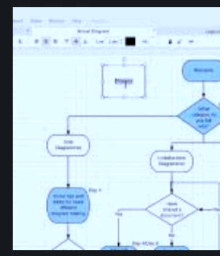
Template: Algorithm Flow

Lucidchart

Visit | Add to | Collections

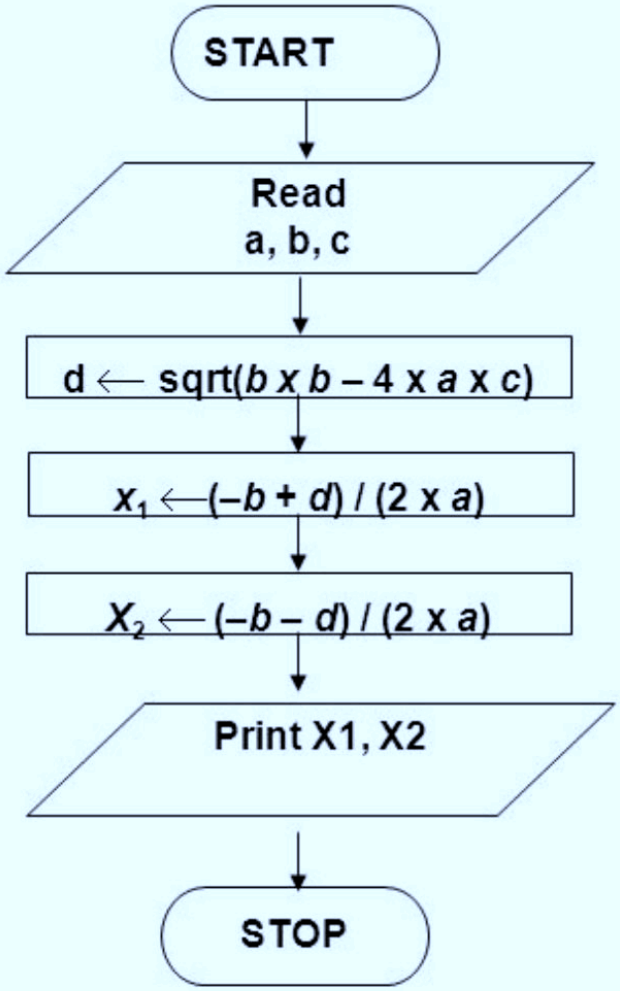
Related images:

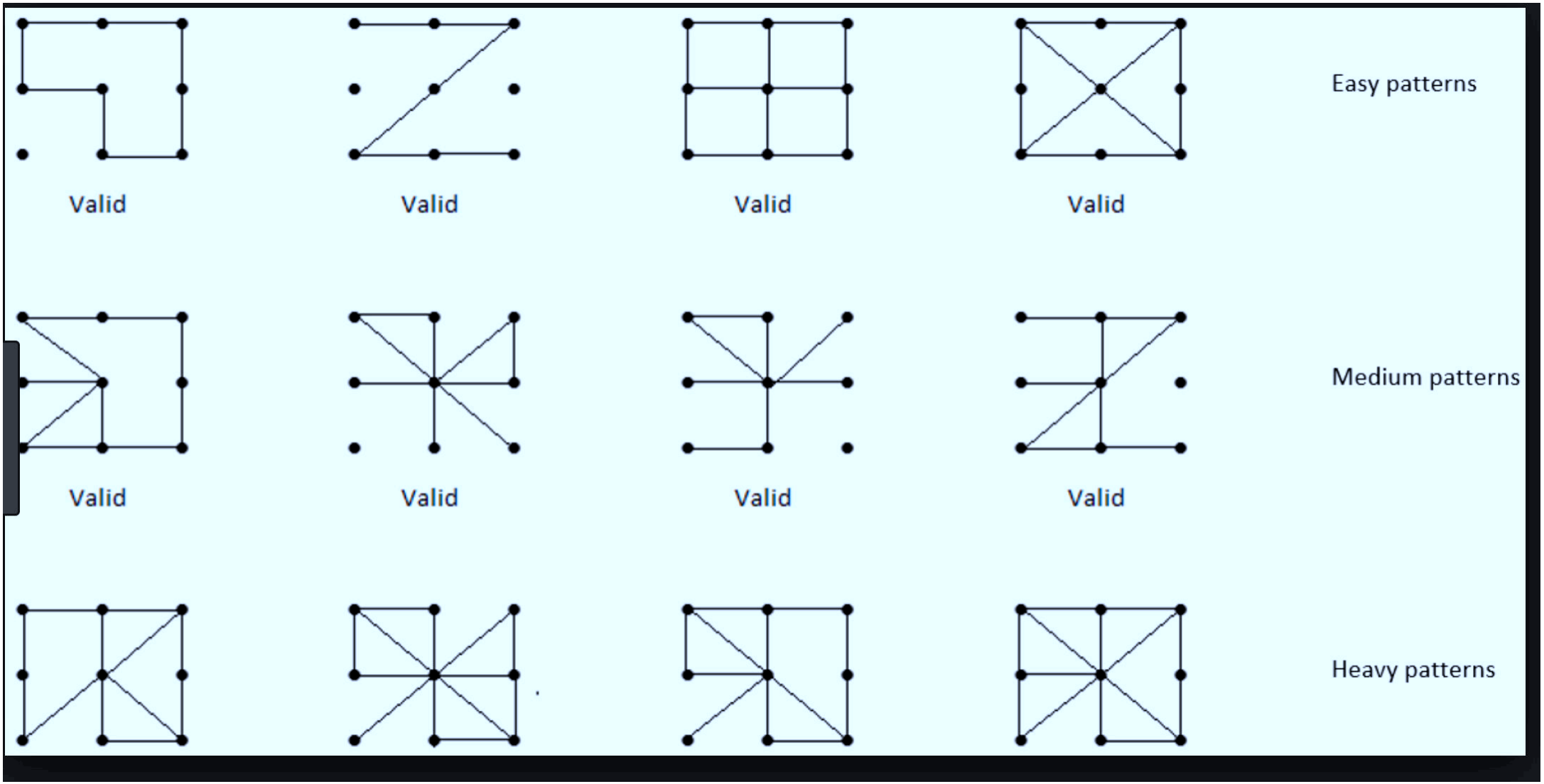




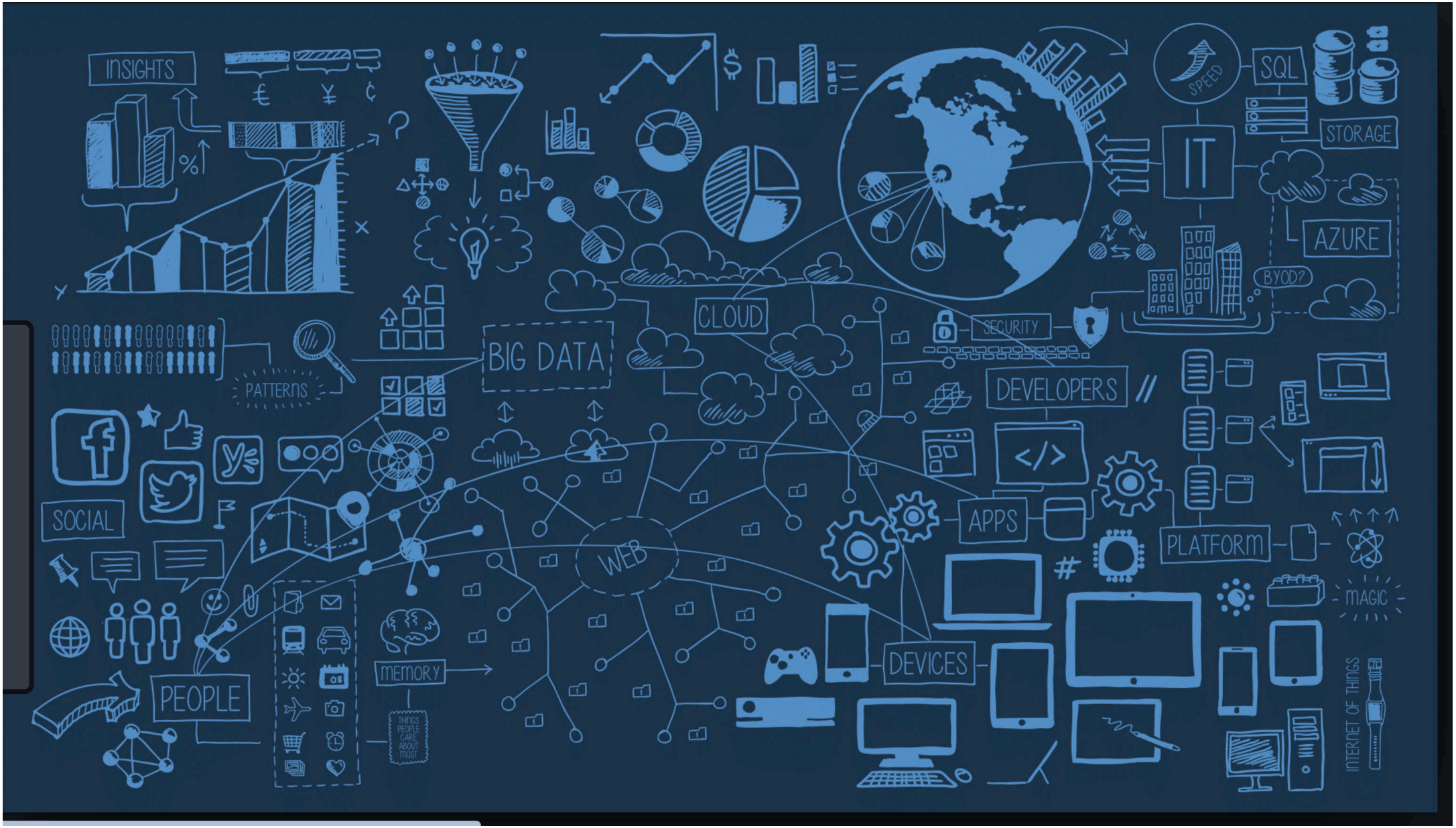



Problem: Write Algorithm and Flowchart to find solution of Quadratic equation

- **Algorithm:**
  - Step 1: Start
  - Step 2: Read a, b, c
  - Step 3: $d \leftarrow \text{sqrt}(\; b \times b - 4 \times a \times c \;)$
  - Step 4: $x1 \leftarrow (-b + d) / (2 \times a)$
  - Step 5: $x2 \leftarrow (-b - d) / (2 \times a)$
  - Step 6: Print $x1$, $x2$
  - Step 7: Stop




Easy patterns
Valid
Valid
Valid
Valid
Medium patterns
Valid
Valid
Valid
Valid
Heavy patterns


INSIGHTS
SPEED
SQL
STORAGE
IT
AZURE
BYOD?
CLOUD
SECURITY
BIG DATA
PATTERNS
DEVELOPERS
SOCIAL
APPS
PLATFORM
WEB
MAGIC
DEVICES
PEOPLE
MEMORY
THINGS PEOPLE CARE ABOUT MOST
INTERNET OF THINGS

Step 1: Start

Step 2: Create a variable to receive the user's email address

Step 3: Clear the variable in case it's not empty

Step 4: Ask the user for an email address

Step 5: Store the response in the variable

Step 6: Check the stored response to see if it is a valid email address

Step 7: Not valid? Go back to Step 3.

Step 8: End

- It should be quantitative if at all possible.
- If there is a clear mathematical relationship between the Independent Variable (IV) and the Dependent Variable (DV) this needs to be stated in your hypothesis.
  - A scientific justification of your hypothesis
  - You must show that you have done some research on the subject through the justification of your hypothesis.
  - You will need to provide reasons for why you think your hypothesis will be the outcome – this is not a personal view, it is a view based on sound scientific judgments based on your research.
  - You should cite sources listed in your bibliography. The justification (scientific explanation) may require discussion of equations, general principles, laws and published examples.

VARIABLES:

Variables are factors that may affect the outcome of your experiment. They are measurable factors, not pieces of equipment. Do not use the word "Amount". It is not specific enough – terms like mass or volume are better.

1. **Independent variable:** Independent Variables are changes that occur in an experiment that are directly caused by the experimenter (you.).
2. **Dependent variable:** Dependent Variables are changes that occur due to independent variables. It is what you are measuring or trying to find out.
3. **Controlled variables:** A Controlled Variable is anything else that could influence the dependent variables..
4. **Uncontrolled variables:** Usually climate factors that you try to keep the same for each sample.

For example, we can set up an experiment in which we measure time needed for an apple to fall from different heights. First we go to the first floor, release the apple and measure the falling time. Then we go one floor up, repeat the measurement, and so on. In this experiment, the height from which we were releasing the apple, we can consider as independent variable, and the falling time as dependent variable, because the time depends on our freely selected height

A clearly laid out section on variables

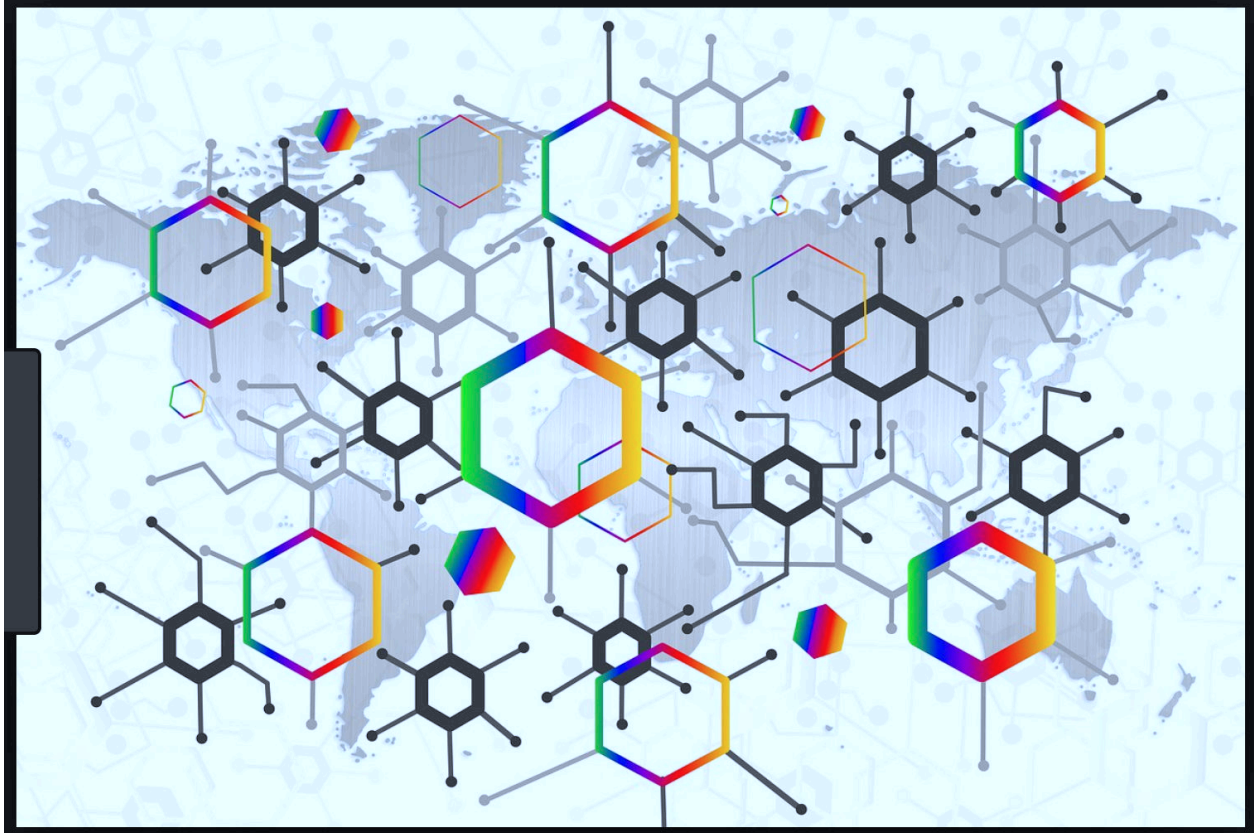
This is where you have the greatest potential to lose marks in the Planning Criterion. Two of the 3 aspects are focused on variables.

Clustering algorithms

towardsdatascience.com

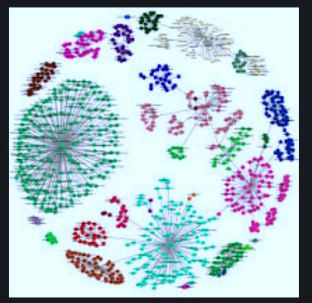
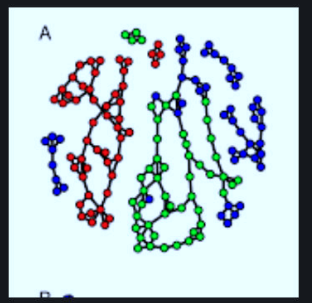
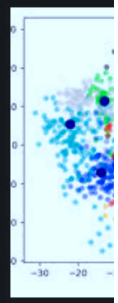
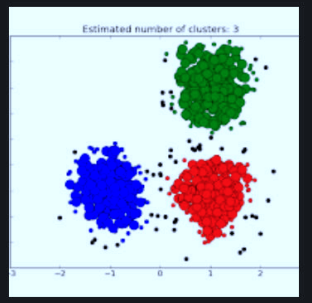
Context In today's competitive world, customer behavior and categorize cu demography and...

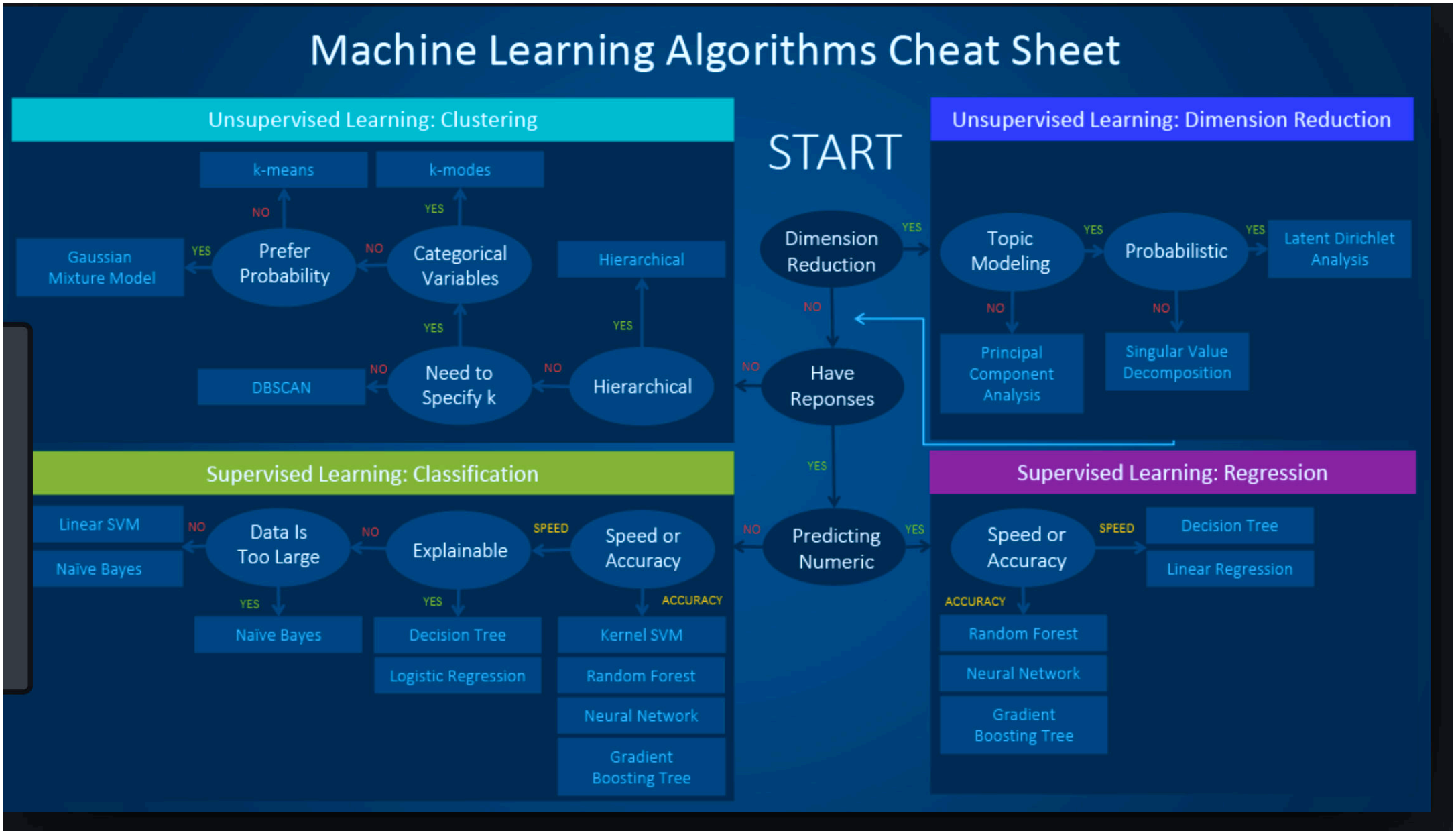
Visit | Add to | Collections



1280 × 853 - Images may be subject to copyright. Learn More

Related images:



Machine Learning Algorithms Cheat Sheet
START
Unsupervised Learning: Clustering
k-means
k-modes
NO
YES
Gaussian Mixture Model
YES
Prefer Probability
NO
Categorical Variables
Hierarchical
YES
YES
DBSCAN
NO
Need to Specify k
NO
Hierarchical
Unsupervised Learning: Dimension Reduction
Dimension Reduction
YES
Topic Modeling
YES
Probabilistic
YES
Latent Dirichlet Analysis
NO
NO
NO
Principal Component Analysis
Singular Value Decomposition
NO
Have Reponses
YES
Supervised Learning: Classification
Linear SVM
NO
Data Is Too Large
NO
Explainable
SPEED
Speed or Accuracy
Naïve Bayes
YES
YES
ACCURACY
Naïve Bayes
Decision Tree
Logistic Regression
Kernel SVM
Random Forest
Neural Network
Gradient Boosting Tree
NO
Predicting Numeric
YES
Supervised Learning: Regression
Speed or Accuracy
SPEED
Decision Tree
Linear Regression
ACCURACY
Random Forest
Neural Network
Gradient Boosting Tree

Machine Learning

HEALTHCARE:
Patient Diagnosis

FINANCE:
Fraud Detection

MANUFACTURING:
Anomaly Detection

RETAIL:
Inventory Optimization

GOVERNMENT:
Smarter Services

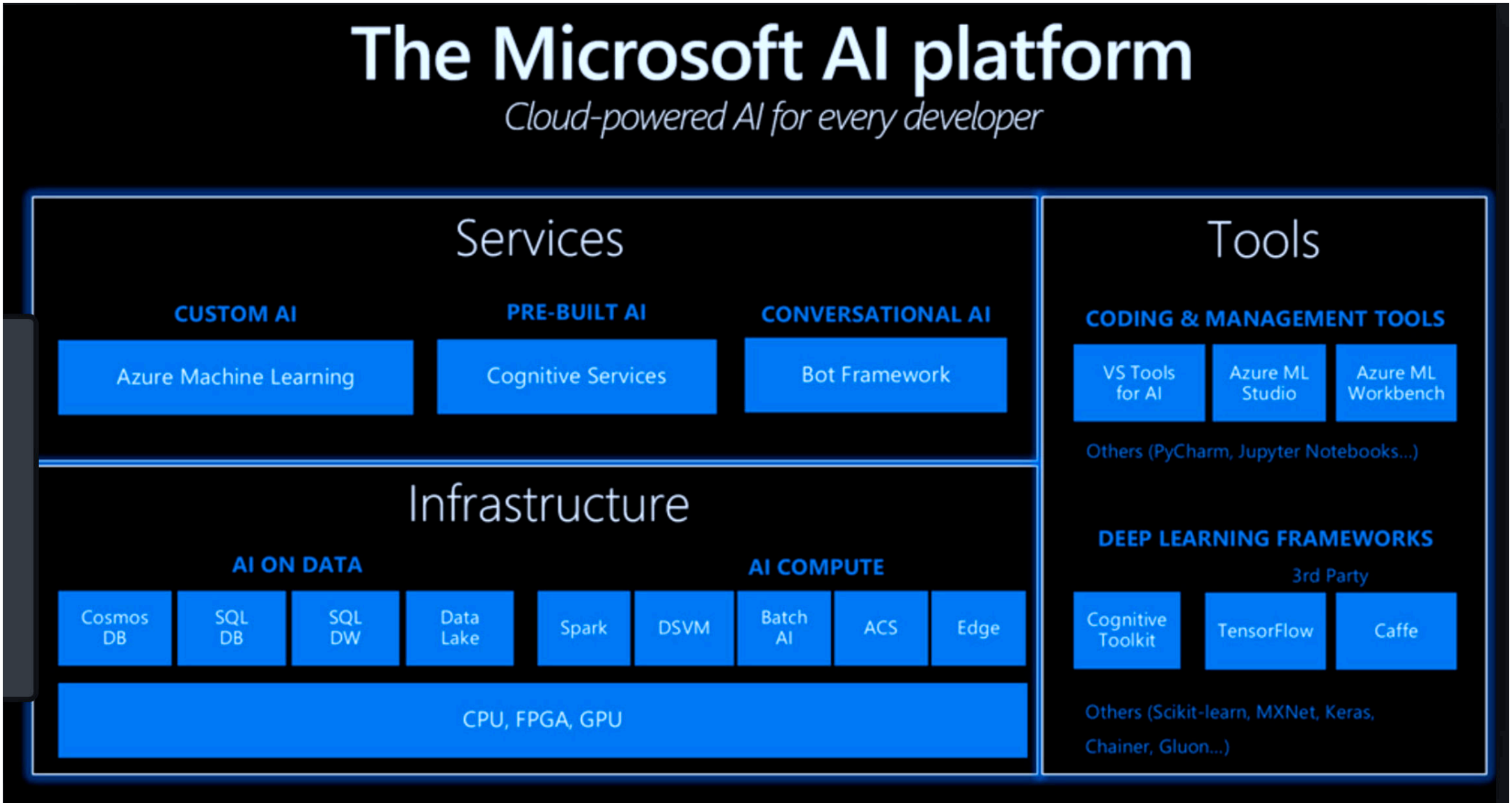
TRANSPORTATION:
Demand Forecasting

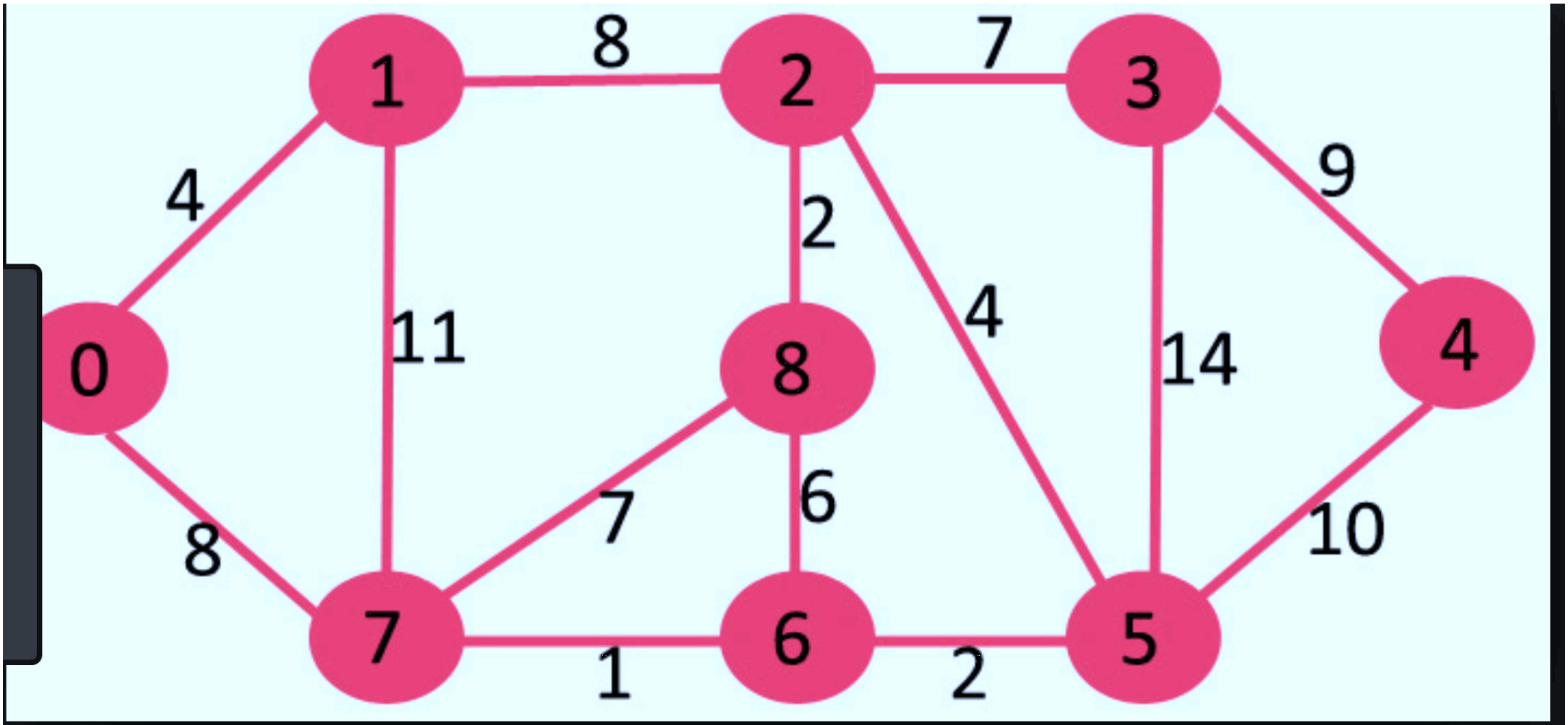
NETWORKS:
Intrusion Detection

E-COMMERCE:
Recommender Systems

MEDIA:
Interaction & Speed

EDUCATION:
Research Insight


The Microsoft AI platform
Cloud-powered AI for every developer
Services
CUSTOM AI
Azure Machine Learning
PRE-BUILT AI
Cognitive Services
CONVERSATIONAL AI
Bot Framework
Infrastructure
AI ON DATA
Cosmos DB
SQL DB
SQL DW
Data Lake
AI COMPUTE
Spark
DSVM
Batch AI
ACS
Edge
CPU, FPGA, GPU
Tools
CODING & MANAGEMENT TOOLS
VS Tools for AI
Azure ML Studio
Azure ML Workbench
Others (PyCharm, Jupyter Notebooks...)
DEEP LEARNING FRAMEWORKS
3rd Party
Cognitive Toolkit
TensorFlow
Caffe
Others (Scikit-learn, MXNet, Keras, Chainer, Gluon...)



Boruvka's algorithm

- Algorithm types we will consider include:
  - Simple recursive algorithms
  - Backtracking algorithms
  - Divide and conquer algorithms
  - Dynamic programming algorithms
  - Greedy algorithms
  - Branch and bound algorithms
  - Brute force algorithms
  - Randomized algorithms

A system that uses Banker's-Algorithm deadlock avoidance has five processes (1, 2, 3,4, and 5) and uses resources of four different types (A, B, C, and D). There are multiple resources of each type. Is the state of the system SAFE? Explain your answer. If the system is safe, show how all the processes could complete their execution successfully. If the system is unsafe, show how deadlock might occur.

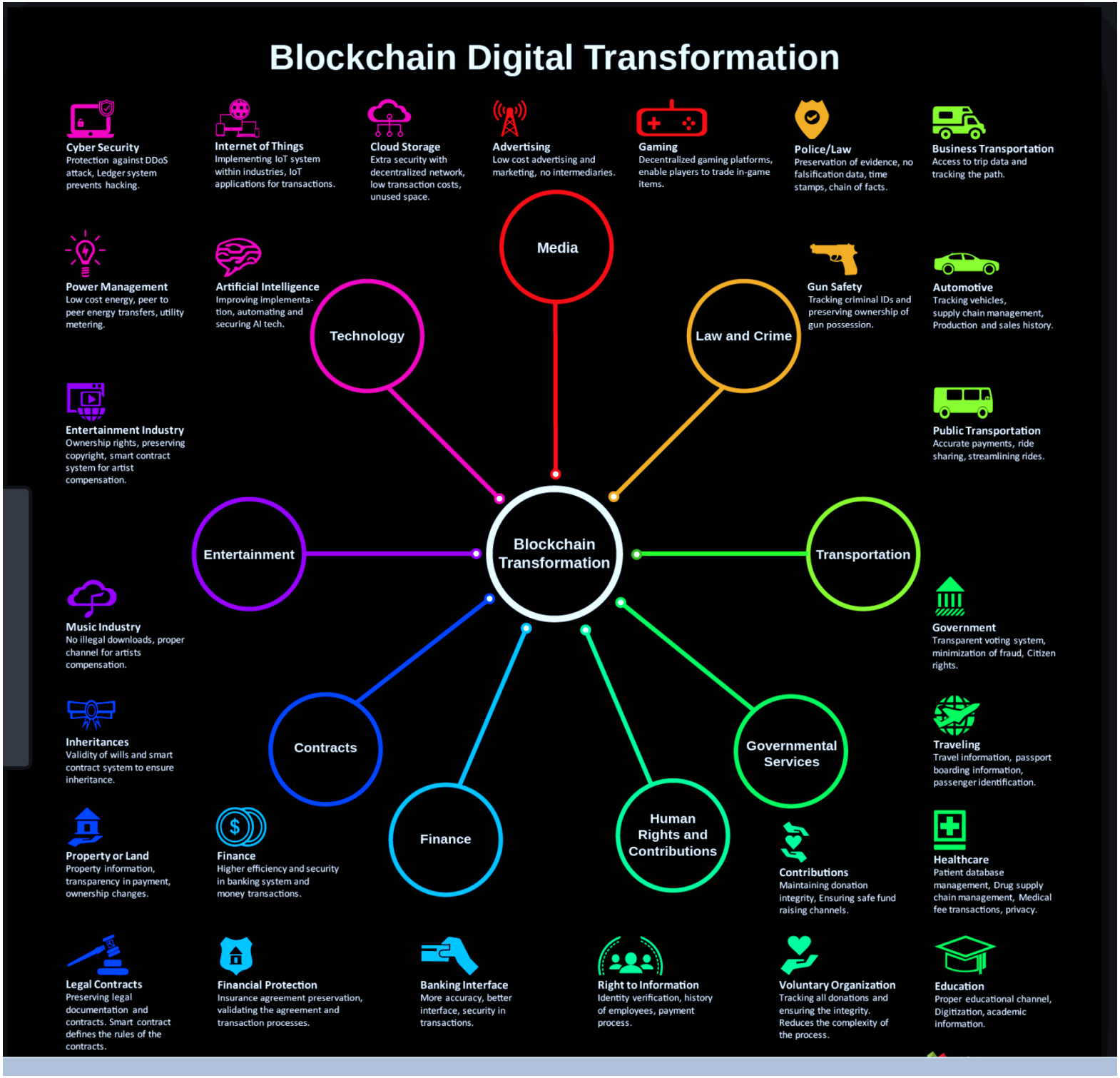
	allocation				Max request				need			
	A	B	C	D	A	B	C	D	A	B	C	D
1	1	0	2	0	3	2	4	2	2	2	2	2
2	0	3	1	2	3	5	1	2	3	2	0	0
3	2	4	5	1	2	7	7	5	0	3	2	4
4	3	0	0	6	5	5	0	8	2	5	0	2
5	4	2	1	3	6	2	1	4	2	0	0	1

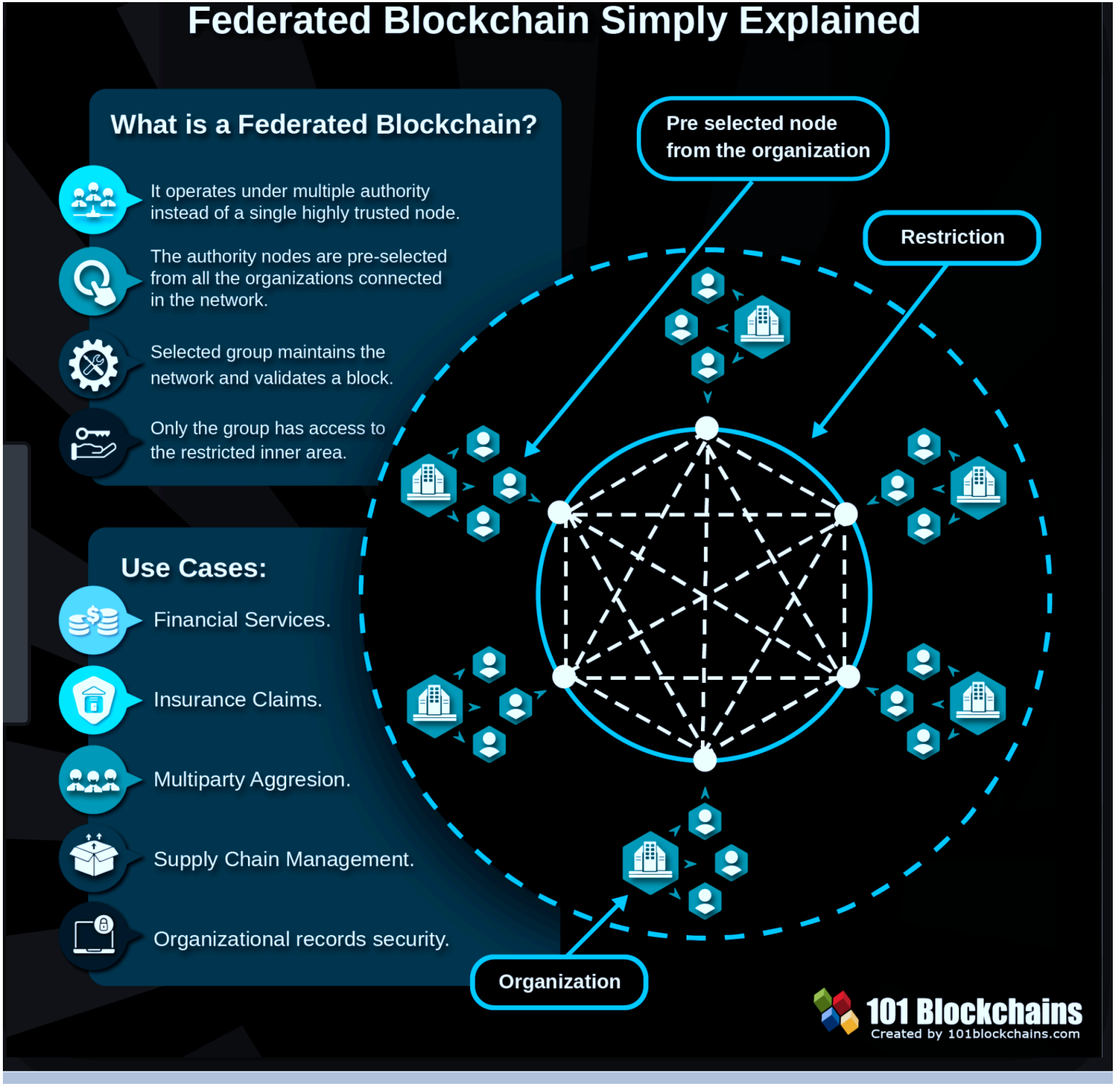
Total Resources

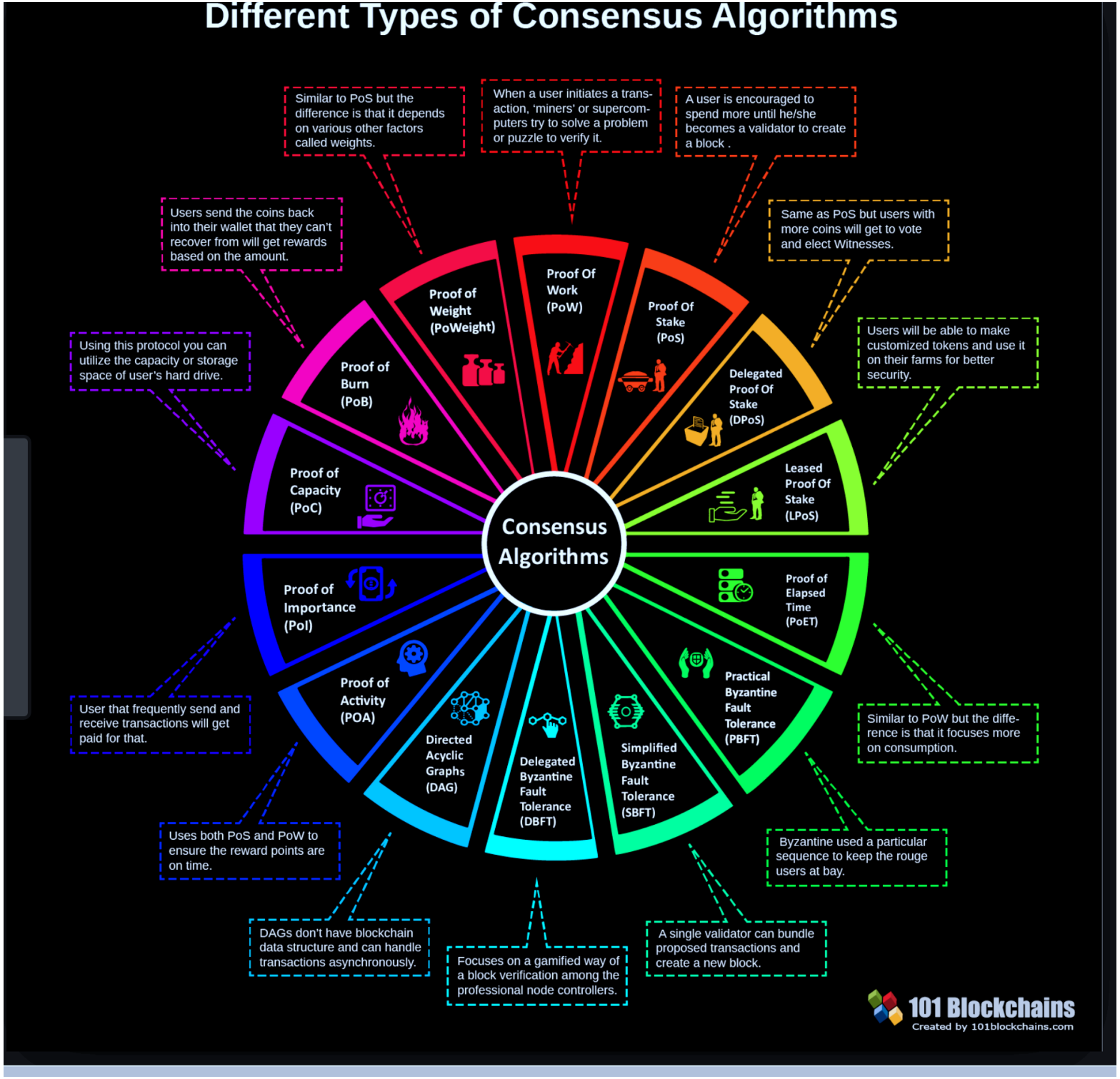
A	B	C	D
13	13	9	13

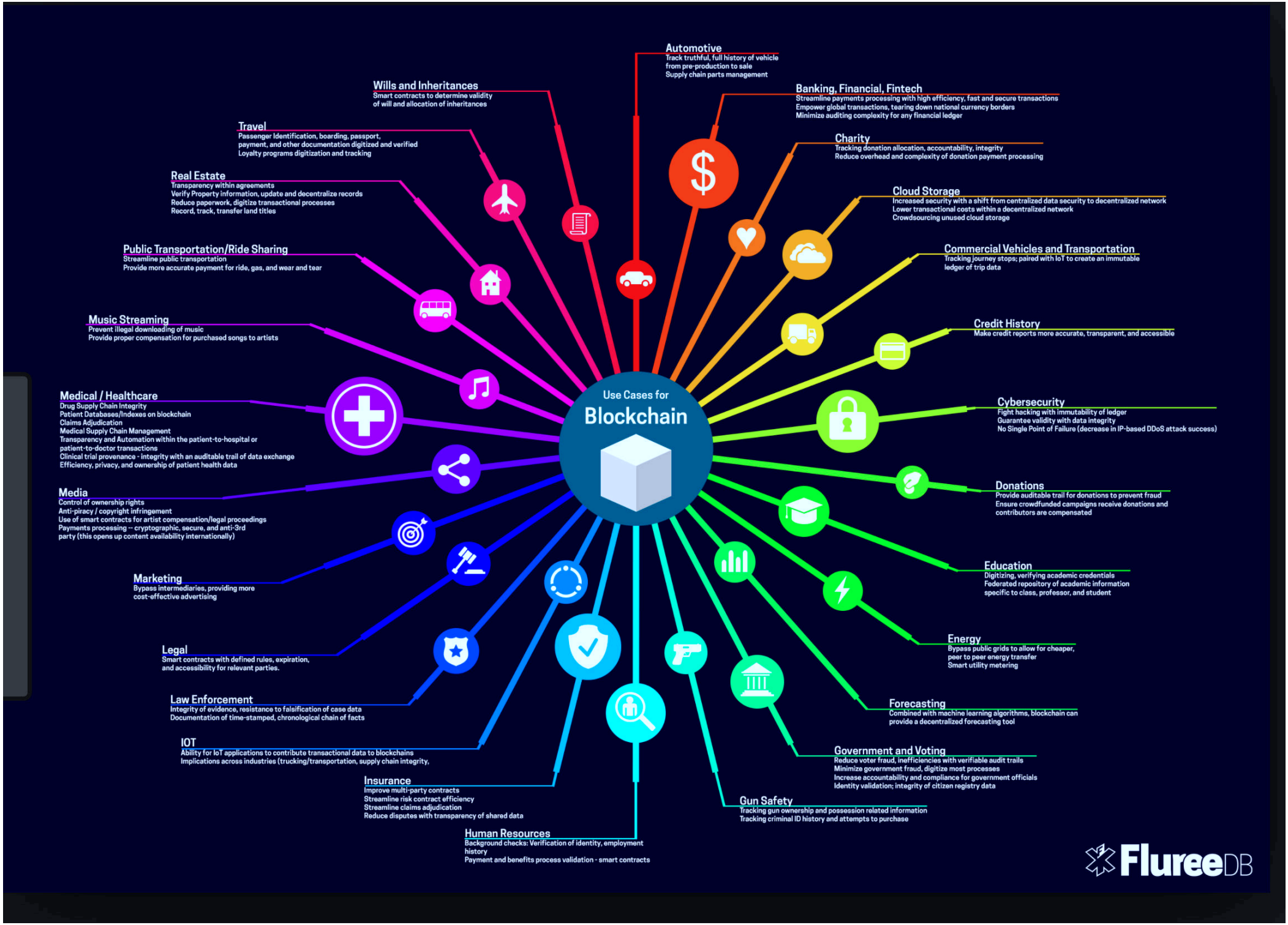
Resources Available

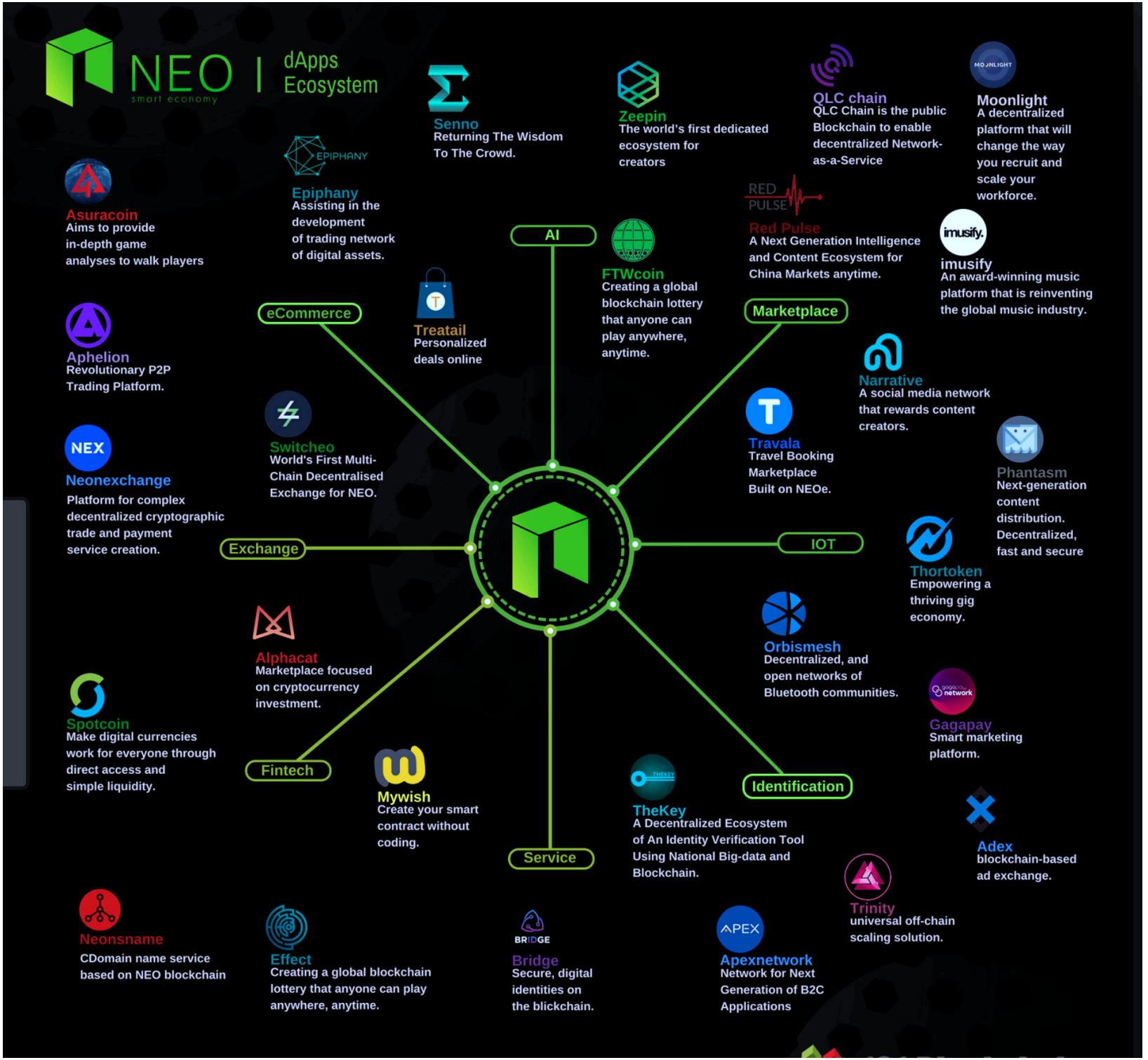
A	B	C	D
3	4	0	1


Blockchain Digital Transformation
Cyber Security
Protection against DDoS attack, Ledger system prevents hacking.
Internet of Things
Implementing IoT system within industries, IoT applications for transactions.
Cloud Storage
Extra security with decentralized network, low transaction costs, unused space.
Advertising
Low cost advertising and marketing, no intermediaries.
Gaming
Decentralized gaming platforms, enable players to trade in-game items.
Police/Law
Preservation of evidence, no falsification data, time stamps, chain of facts.
Business Transportation
Access to trip data and tracking the path.
Power Management
Low cost energy, peer to peer energy transfers, utility metering.
Artificial Intelligence
Improving implementation, automating and securing AI tech.
Media
Technology
Law and Crime
Gun Safety
Tracking criminal IDs and preserving ownership of gun possession.
Automotive
Tracking vehicles, supply chain management, Production and sales history.
Entertainment Industry
Ownership rights, preserving copyright, smart contract system for artist compensation.
Public Transportation
Accurate payments, ride sharing, streamlining rides.
Entertainment
Blockchain Transformation
Transportation
Music Industry
No illegal downloads, proper channel for artists compensation.
Government
Transparent voting system, minimization of fraud, Citizen rights.
Inheritances
Validity of wills and smart contract system to ensure inheritance.
Contracts
Governmental Services
Traveling
Travel information, passport boarding information, passenger identification.
Property or Land
Property information, transparency in payment, ownership changes.
Finance
Higher efficiency and security in banking system and money transactions.
Finance
Human Rights and Contributions
Contributions
Maintaining donation integrity, Ensuring safe fund raising channels.
Healthcare
Patient database management, Drug supply chain management, Medical fee transactions, privacy.
Legal Contracts
Preserving legal documentation and contracts. Smart contract defines the rules of the contracts.
Financial Protection
Insurance agreement preservation, validating the agreement and transaction processes.
Banking Interface
More accuracy, better interface, security in transactions.
Right to Information
Identity verification, history of employees, payment process.
Voluntary Organization
Tracking all donations and ensuring the integrity. Reduces the complexity of the process.
Education
Proper educational channel, Digitization, academic information.

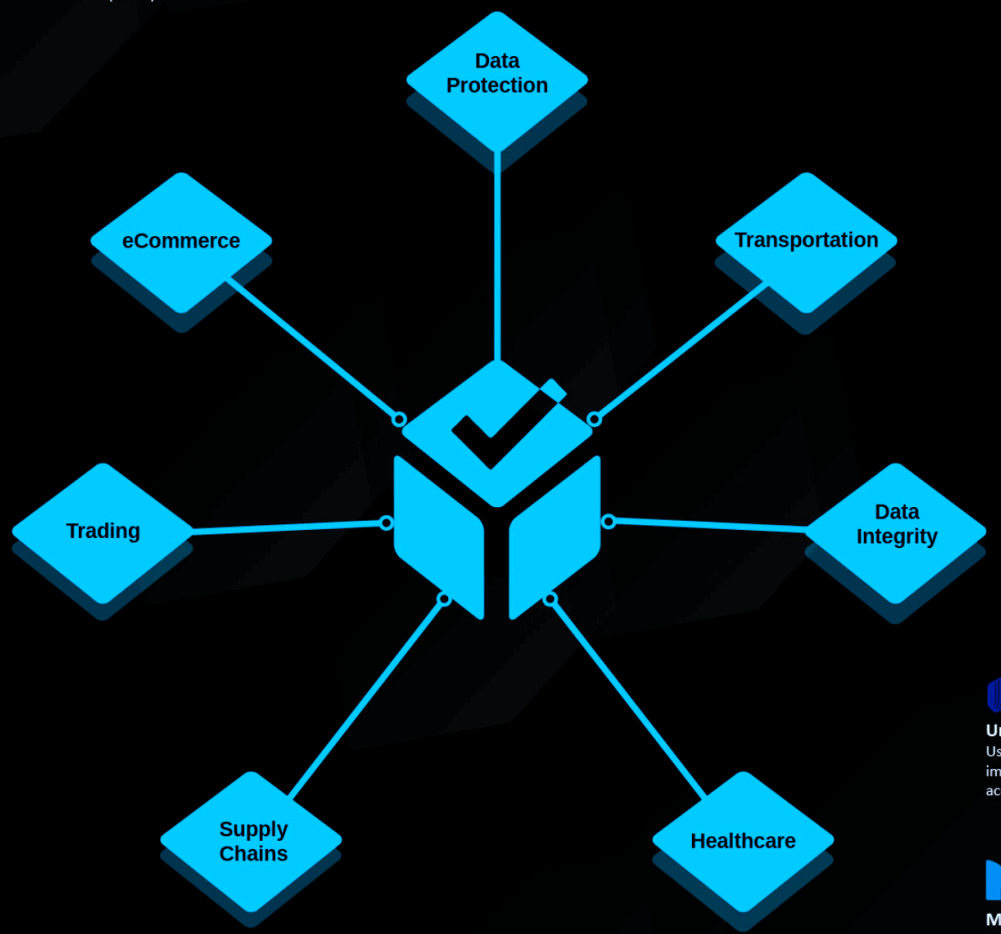

Federated Blockchain Simply Explained
What is a Federated Blockchain?
It operates under multiple authority instead of a single highly trusted node.
The authority nodes are pre-selected from all the organizations connected in the network.
Selected group maintains the network and validates a block.
Only the group has access to the restricted inner area.
Use Cases:
Financial Services.
Insurance Claims.
Multiparty Aggresion.
Supply Chain Management.
Organizational records security.
Pre selected node from the organization
Restriction
Organization
101 Blockchains
Created by 101blockchains.com


Different Types of Consensus Algorithms
Consensus Algorithms
Proof of Work (PoW)
When a user initiates a transaction, 'miners' or supercomputers try to solve a problem or puzzle to verify it.
Proof of Weight (PoWeight)
Similar to PoS but the difference is that it depends on various other factors called weights.
Proof of Stake (PoS)
A user is encouraged to spend more until he/she becomes a validator to create a block .
Delegated Proof Of Stake (DPoS)
Same as PoS but users with more coins will get to vote and elect Witnesses.
Leased Proof Of Stake (LPoS)
Users will be able to make customized tokens and use it on their farms for better security.
Proof of Elapsed Time (PoET)
Similar to PoW but the difference is that it focuses more on consumption.
Practical Byzantine Fault Tolerance (PBFT)
Byzantine used a particular sequence to keep the rouge users at bay.
Simplified Byzantine Fault Tolerance (SBFT)
A single validator can bundle proposed transactions and create a new block.
Delegated Byzantine Fault Tolerance (DBFT)
Focuses on a gamified way of a block verification among the professional node controllers.
Directed Acyclic Graphs (DAG)
DAGs don't have blockchain data structure and can handle transactions asynchronously.
Proof of Activity (POA)
Uses both PoS and PoW to ensure the reward points are on time.
Proof of Importance (PoI)
User that frequently send and receive transactions will get paid for that.
Proof of Capacity (PoC)
Using this protocol you can utilize the capacity or storage space of user's hard drive.
Proof of Burn (PoB)
Users send the coins back into their wallet that they can't recover from will get rewards based on the amount.
101 Blockchains
Created by 101blockchains.com


Use Cases for
Blockchain
Automotive
Track truthful, full history of vehicle from pre-production to sale
Supply chain parts management
Wills and Inheritances
Smart contracts to determine validity of will and allocation of inheritances
Banking, Financial, Fintech
Streamline payments processing with high efficiency, fast and secure transactions
Empower global transactions, tearing down national currency borders
Minimize auditing complexity for any financial ledger
Travel
Passenger Identification, boarding, passport, payment, and other documentation digitized and verified
Loyalty programs digitization and tracking
Charity
Tracking donation allocation, accountability, integrity
Reduce overhead and complexity of donation payment processing
Real Estate
Transparency within agreements
Verify Property information, update and decentralize records
Reduce paperwork, digitize transactional processes
Record, track, transfer land titles
Cloud Storage
Increased security with a shift from centralized data security to decentralized network
Lower transactional costs within a decentralized network
Crowdsourcing unused cloud storage
Public Transportation/Ride Sharing
Streamline public transportation
Provide more accurate payment for ride, gas, and wear and tear
Commercial Vehicles and Transportation
Tracking journey stops; paired with IoT to create an immutable ledger of trip data
Music Streaming
Prevent illegal downloading of music
Provide proper compensation for purchased songs to artists
Credit History
Make credit reports more accurate, transparent, and accessible
Medical / Healthcare
Drug Supply Chain Integrity
Patient Databases/Indexes on blockchain
Claims Adjudication
Medical Supply Chain Management
Transparency and Automation within the patient-to-hospital or patient-to-doctor transactions
Clinical trial provenance - integrity with an auditable trail of data exchange
Efficiency, privacy, and ownership of patient health data
Cybersecurity
Fight hacking with immutability of ledger
Guarantee validity with data integrity
No Single Point of Failure (decrease in IP-based DDoS attack success)
Media
Control of ownership rights
Anti-piracy / copyright infringement
Use of smart contracts for artist compensation/legal proceedings
Payments processing -- cryptographic, secure, and anti-3rd party (this opens up content availability internationally)
Donations
Provide auditable trail for donations to prevent fraud
Ensure crowdfunded campaigns receive donations and contributors are compensated
Marketing
Bypass intermediaries, providing more cost-effective advertising
Education
Digitizing, verifying academic credentials
Federated repository of academic information specific to class, professor, and student
Energy
Bypass public grids to allow for cheaper, peer to peer energy transfer
Smart utility metering
Legal
Smart contracts with defined rules, expiration, and accessibility for relevant parties.
Forecasting
Combined with machine learning algorithms, blockchain can provide a decentralized forecasting tool
Law Enforcement
Integrity of evidence, resistance to falsification of case data
Documentation of time-stamped, chronological chain of facts
Government and Voting
Reduce voter fraud, inefficiencies with verifiable audit trails
Minimize government fraud, digitize most processes
Increase accountability and compliance for government officials
Identity validation; integrity of citizen registry data
IOT
Ability for IoT applications to contribute transactional data to blockchains
Implications across industries (trucking/transportation, supply chain integrity,
Insurance
Improve multi-party contracts
Streamline risk contract efficiency
Streamline claims adjudication
Reduce disputes with transparency of shared data
Gun Safety
Tracking gun ownership and possession related information
Tracking criminal ID history and attempts to purchase
Human Resources
Background checks: Verification of identity, employment history
Payment and benefits process validation - smart contracts
FlureeDB


NEO | dApps Ecosystem
smart economy
Senno
Returning The Wisdom To The Crowd.
Zeepin
The world's first dedicated ecosystem for creators
QLC chain
QLC Chain is the public Blockchain to enable decentralized Network-as-a-Service
Moonlight
A decentralized platform that will change the way you recruit and scale your workforce.
Epiphany
Assisting in the development of trading network of digital assets.
Asuracoin
Aims to provide in-depth game analyses to walk players
AI
FTWcoin
Creating a global blockchain lottery that anyone can play anywhere, anytime.
Red Pulse
A Next Generation Intelligence and Content Ecosystem for China Markets anytime.
imusify
An award-winning music platform that is reinventing the global music industry.
eCommerce
Treatail
Personalized deals online
Marketplace
Aphelion
Revolutionary P2P Trading Platform.
Narrative
A social media network that rewards content creators.
Switcheo
World's First Multi-Chain Decentralised Exchange for NEO.
Travala
Travel Booking Marketplace Built on NEOe.
Phantasm
Next-generation content distribution. Decentralized, fast and secure
NEX
Neonexchange
Platform for complex decentralized cryptographic trade and payment service creation.
Exchange
IOT
Thortoken
Empowering a thriving gig economy.
Alphacat
Marketplace focused on cryptocurrency investment.
Orbismesh
Decentralized, and open networks of Bluetooth communities.
Spotcoin
Make digital currencies work for everyone through direct access and simple liquidity.
Gagapay
Smart marketing platform.
Fintech
Mywish
Create your smart contract without coding.
Identification
TheKey
A Decentralized Ecosystem of An Identity Verification Tool Using National Big-data and Blockchain.
Adex
blockchain-based ad exchange.
Service
Trinity
universal off-chain scaling solution.
Neonsname
CDomain name service based on NEO blockchain
Effect
Creating a global blockchain lottery that anyone can play anywhere, anytime.
Bridge
Secure, digital identities on the blickchain.
Apexnetwork
Network for Next Generation of B2C Applications

Enterprises Which Are Implementing Blockchain Technology



Apple
Patented blockchain technology for time stamping data.

Facebook
Exploring the use of blockchain to enhance data security and users privacy.

Google
Exploring the use of blockchain technology to enhance cloud service security and for data protection.

Baidu
Using blockchain to enhance intellectual rights management.

Ford
Leveraging blockchain technology to enhance the mobility of technologies.

Tencent
A Solution for verifying invoice authenticity and for ensuring tax compliance.

Toyota
Planning to use blockchain technology to enhance autonomous driving technology.

Alibaba
Using blockchain technology to track luxury goods in its e-commerce platforms.

Bristish Airways
Implementing blockchain to manage flight data as well as verifying travelers identity.

Prudential
Unveils a blockchain powered trading platform for small and medium-sized enterprises.

AIA Group
Launched the first of its kind bancassurance for sharing policy data.

BHP Billiton
Leveraging blockchain technology for supply chains management.

UnitedHealthcare
Using blockchain technology to improve doctors directories to enable accurate insurance claim fillings.

FedEx
Working on blockchain solution for settling customer disputes.

Metlife
Using blockchain technology for storing patients medical records for insurance purposes.

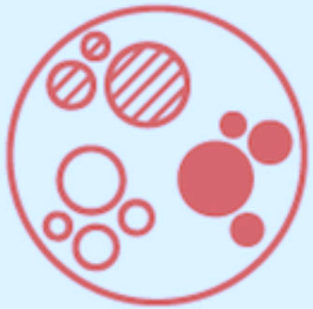
Nestle
Using blockchain technology in supply management to track baby food products.

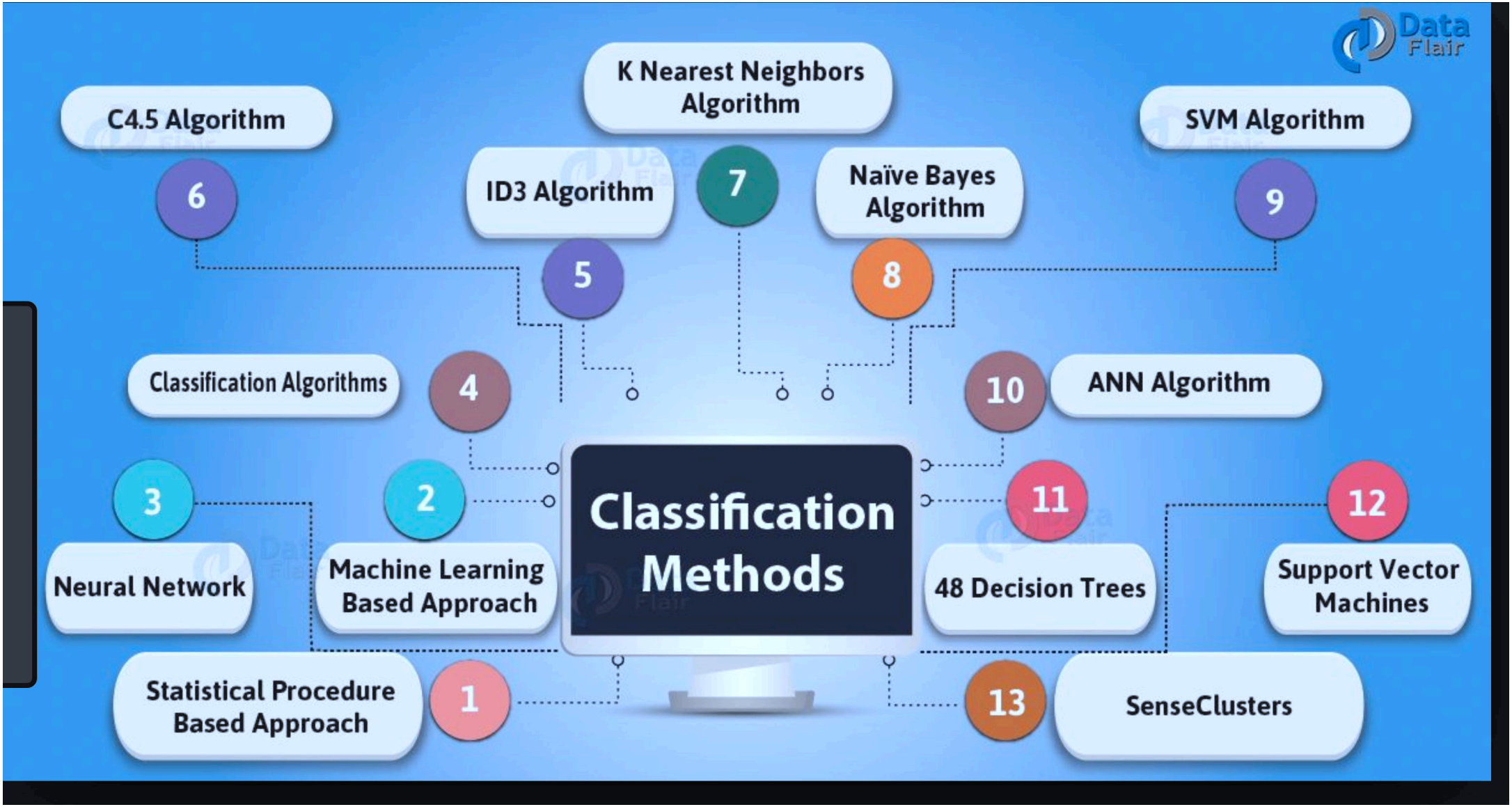
Maersk
Blockchain system for tracking movement of shipments between ports.

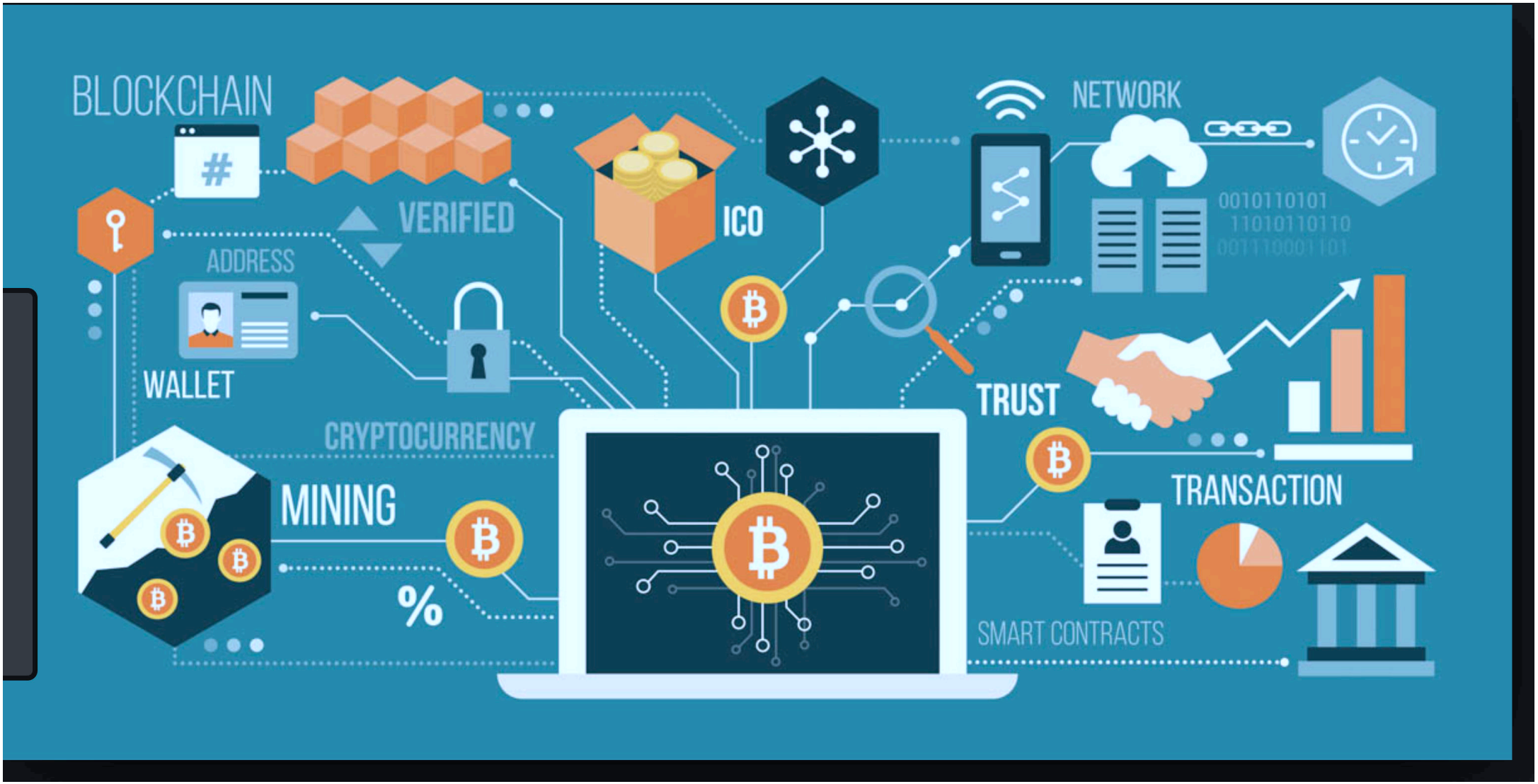
UPS
Blockchain powered logistics monitoring and management solution.

Samsung
Intends to use blockchain technology to enhance supply chain management when it comes to electronics shipments.

Walmart
Using blockchain technology to track product movement from farmers to stores.

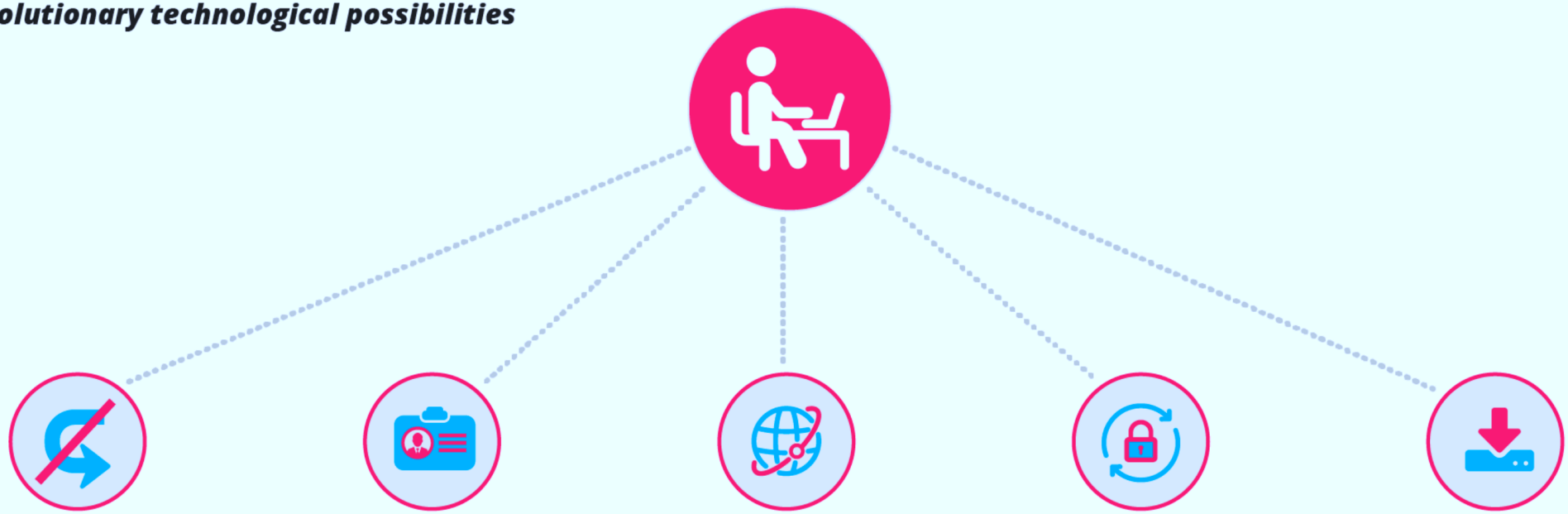
Algorithm Type	Graph Problem	Examples
 **Pathfinding & Search**	Find the optimal path or evaluate route availability and quality	• Find the quickest route to travel from A to B • Telephone call routing
 **Centrality**	Determine the importance of distinct nodes in the networks	• Determine social media influencers • Find likely attack targets in communication and transportation networks
 **Community Detection**	Evaluate how a group is clustered or partitioned	• Segment customers • Find potential members of a fraud ring


Data Flair
Classification Methods
1
Statistical Procedure Based Approach
2
Machine Learning Based Approach
3
Neural Network
4
Classification Algorithms
5
ID3 Algorithm
6
C4.5 Algorithm
7
K Nearest Neighbors Algorithm
8
Naïve Bayes Algorithm
9
SVM Algorithm
10
ANN Algorithm
11
48 Decision Trees
12
Support Vector Machines
13
SenseClusters


BLOCKCHAIN
VERIFIED
ADDRESS
WALLET
ICO
NETWORK
0010110101
11010110110
001110001101
TRUST
CRYPTOCURRENCY
MINING
%
TRANSACTION
SMART CONTRACTS

Blockgeeks

Cryptocurrency opens the door for revolutionary technological possibilities



Irreversible

fter a confirmation a transaction an't be reversed. By nobody. And nobody means nobody. Not you, not your bank, not the president of the United States, not Satoshi, not your miner. Nobody. If you send money, you send it. Period. No one can help you, if you sent your funds to a scammer or if a hacker stole them from your computer. There is no safety net

Pseudonymous

Neither transactions nor accounts are connected to real world identities. You receive Bitcoins on so-called addresses, which are randomly seeming chains of around 30 characters. While it is usually possible to analyze the transaction flow, it is not necessarily possible to connect the real world identity of users with those addresses

Fast and global

Transaction are propagated nearly instantly in the network and are confirmed in a couple of minutes. Since they happen in a global network of computers they are completely indifferent of your physical location. It doesn't matter if I send Bitcoin to my neighbour or to someone on the other side of the world.

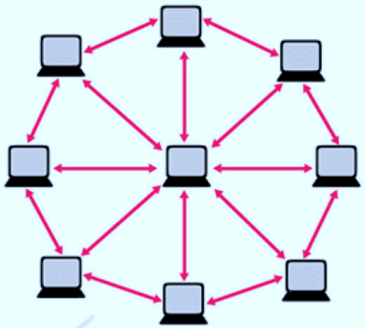
Secure

Cryptocurrency funds are locked in a public key cryptography system. Only the owner of the private key is able to send cryptocurrency. Strong cryptography and the magic of big numbers makes it impossible to break this scheme. A Bitcoin address is more secure than Fort Knox.

Permissionless

You don't have to ask anybody to use cryptocurreny. It's just a software that everybody can download for free. After you installed it, you can receive and send Bitcoins or other cryptocurrency. No one can prevent you. There is no gatekeeper.

Someone requests a transaction.



The requested transaction is broadcast to a P2P network consisting of computers, known as nodes.

Validation

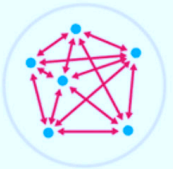
The network of nodes validates the transaction and the user's status using known algorithms.

A verified transaction can involve cryptocurrency, contracts, records, or other information.

cryptocurrency

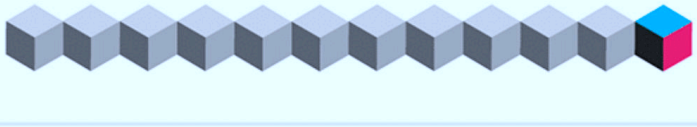
Has no intrinsic value in that is not redeemable for another commodity such as gold.

Has no physical form and exists only in the network.

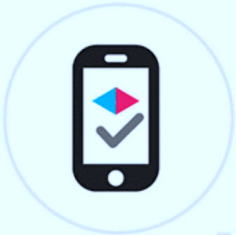


Its supply is not determined by a central bank and the network is completely decentralized.

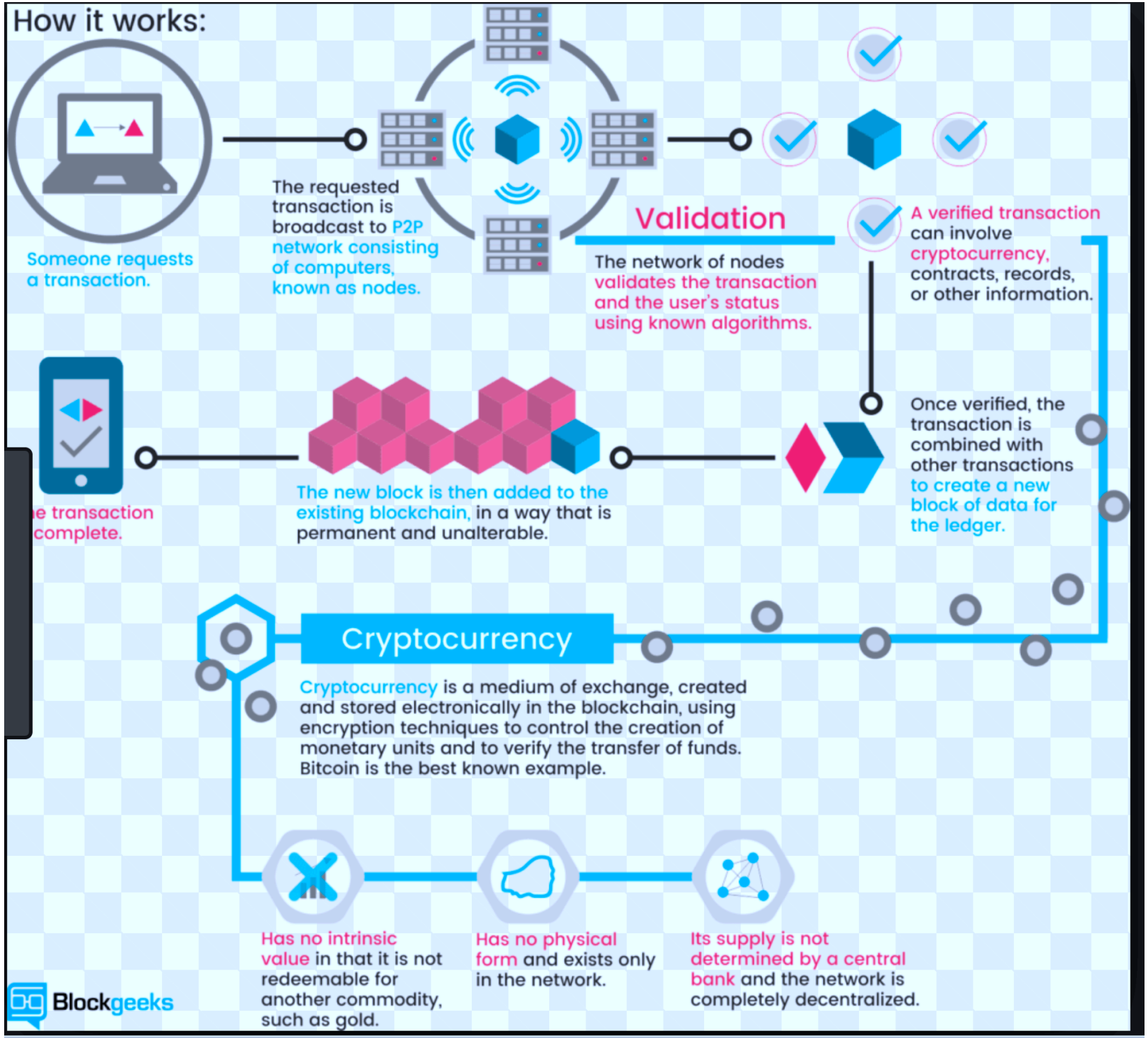
Once verified, the transaction is combined with other transactions to create a new block of data for the ledger,

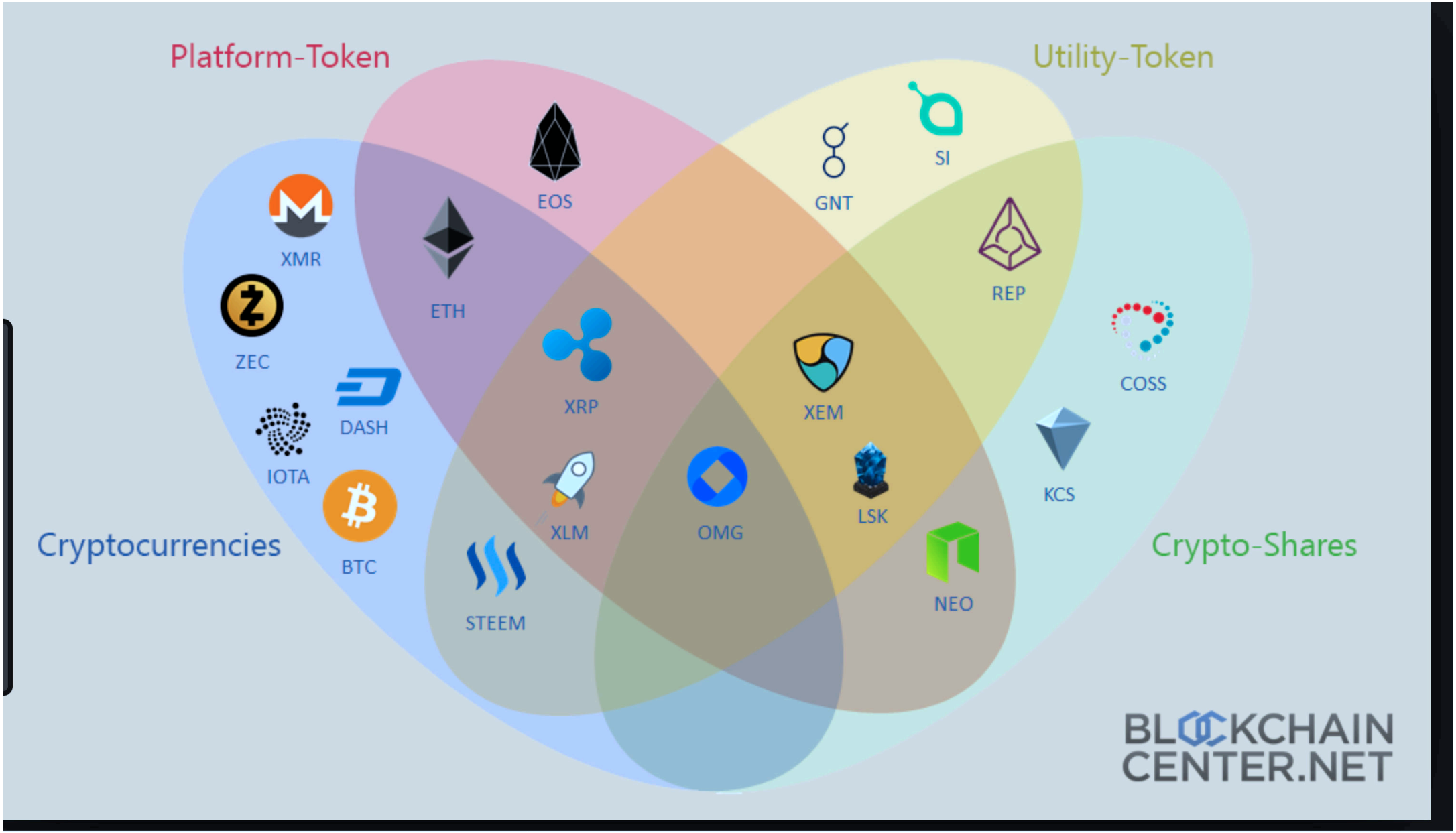


The new block is then added to the existing blockchain, in a way that is permanent and unalterable.



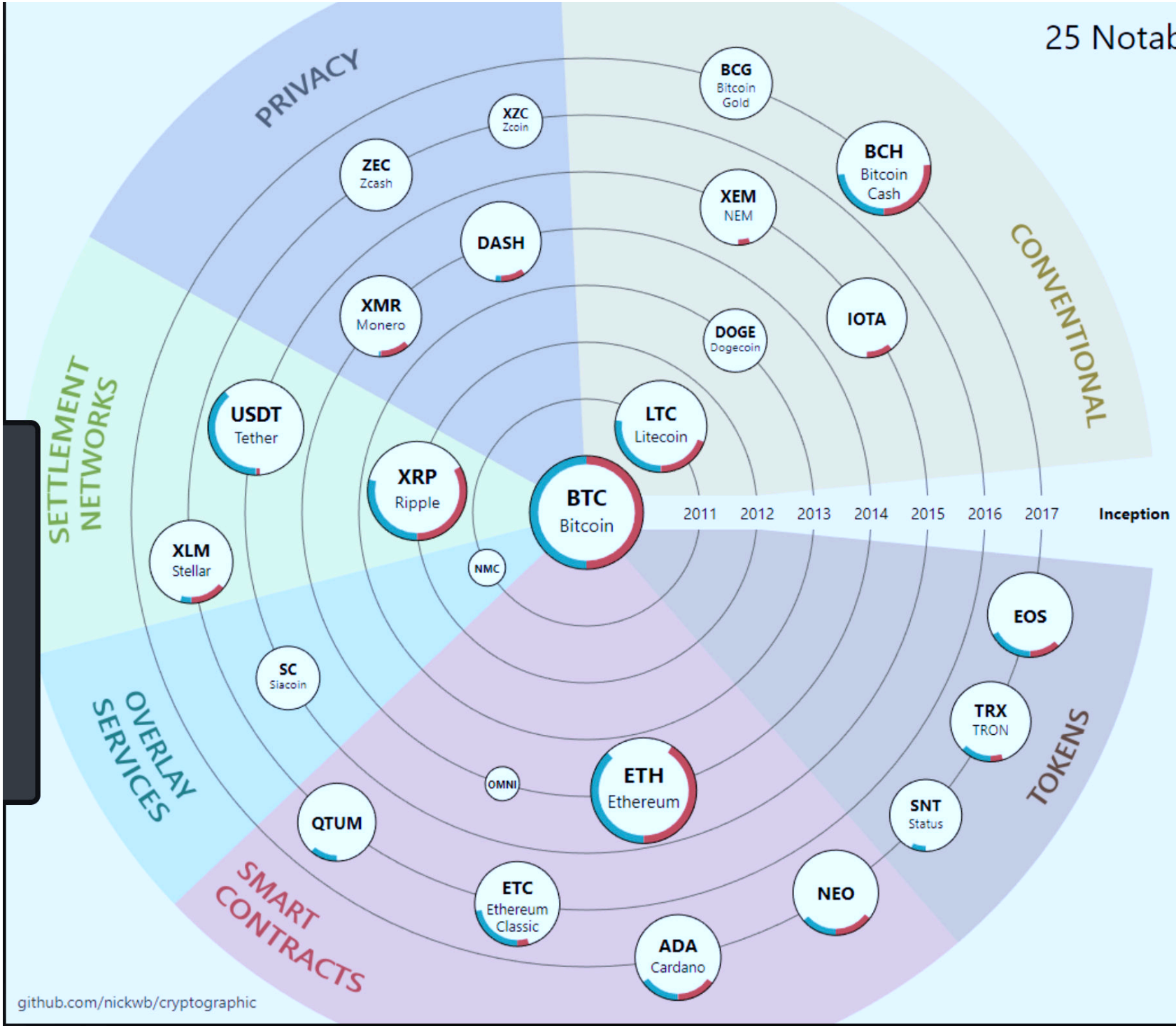
The transaction is complete.


How it works:
Someone requests a transaction.
The requested transaction is broadcast to P2P network consisting of computers, known as nodes.
Validation
The network of nodes validates the transaction and the user's status using known algorithms.
A verified transaction can involve cryptocurrency, contracts, records, or other information.
Once verified, the transaction is combined with other transactions to create a new block of data for the ledger.
The new block is then added to the existing blockchain, in a way that is permanent and unalterable.
e transaction complete.
Cryptocurrency
Cryptocurrency is a medium of exchange, created and stored electronically in the blockchain, using encryption techniques to control the creation of monetary units and to verify the transfer of funds. Bitcoin is the best known example.
Has no intrinsic value in that it is not redeemable for another commodity, such as gold.
Has no physical form and exists only in the network.
Its supply is not determined by a central bank and the network is completely decentralized.
Blockgeeks


Platform-Token
Utility-Token
EOS
GNT
SI
XMR
ETH
REP
ZEC
COSS
XRP
XEM
DASH
KCS
IOTA
XLM
OMG
LSK
BTC
Cryptocurrencies
Crypto-Shares
NEO
STEEM
BLOCKCHAIN
CENTER.NET

25 Notable Cryptocurrencies

A visualization by nickwb



Top 5 by Market Capitalization

Bitcoin:	USD	194 billion
Ethereum:	USD	85 billion
Ripple:	USD	41 billion
Bitcoin Cash:	USD	22 billion
Litecoin:	USD	12 billion

Top 5 by 30 Day Trade Volume

Bitcoin:	USD	230 billion
Tether:	USD	81 billion
Ethereum:	USD	76 billion
Ripple:	USD	32 billion
Litecoin:	USD	28 billion

Key

CODE
Name

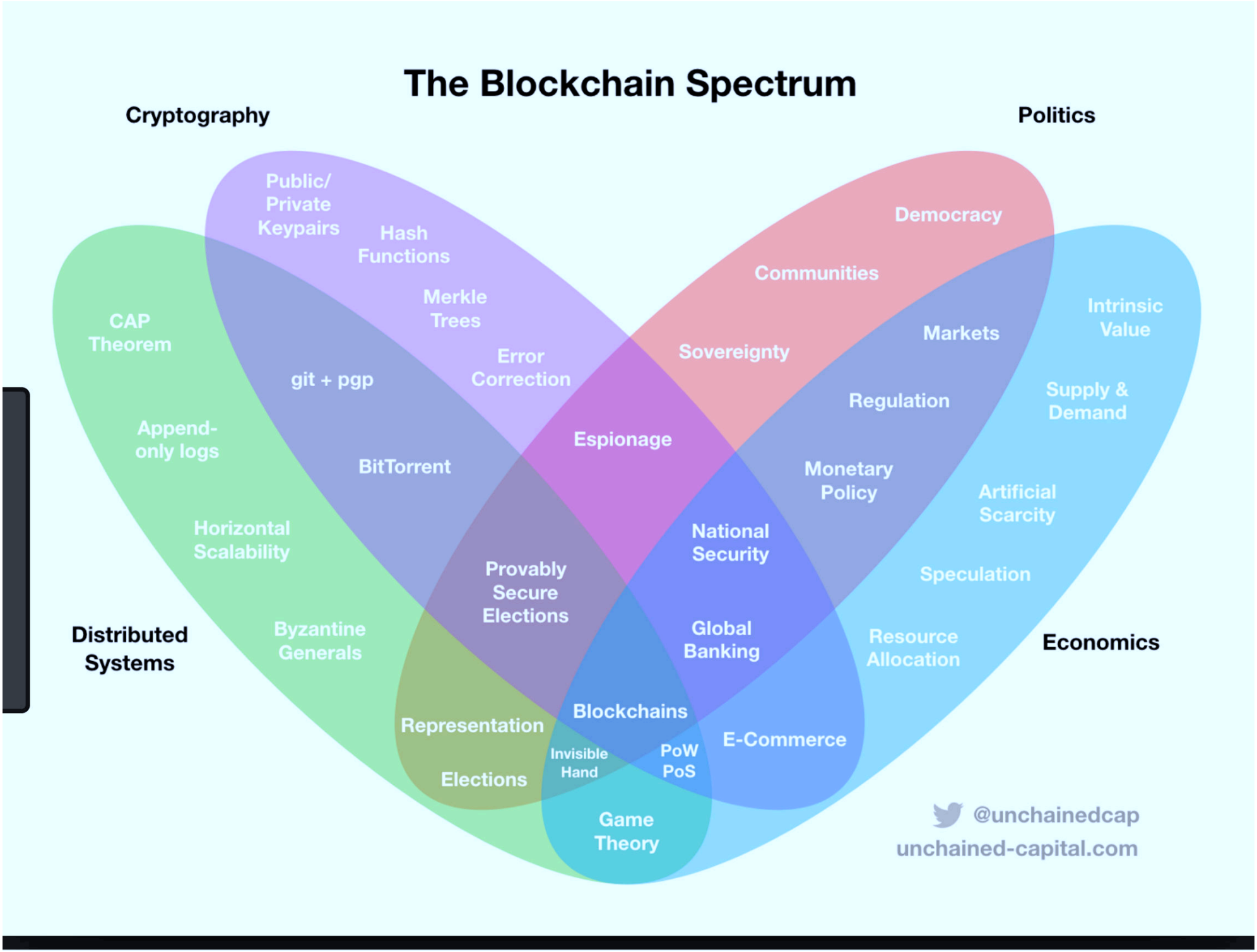
Relative market capitalization

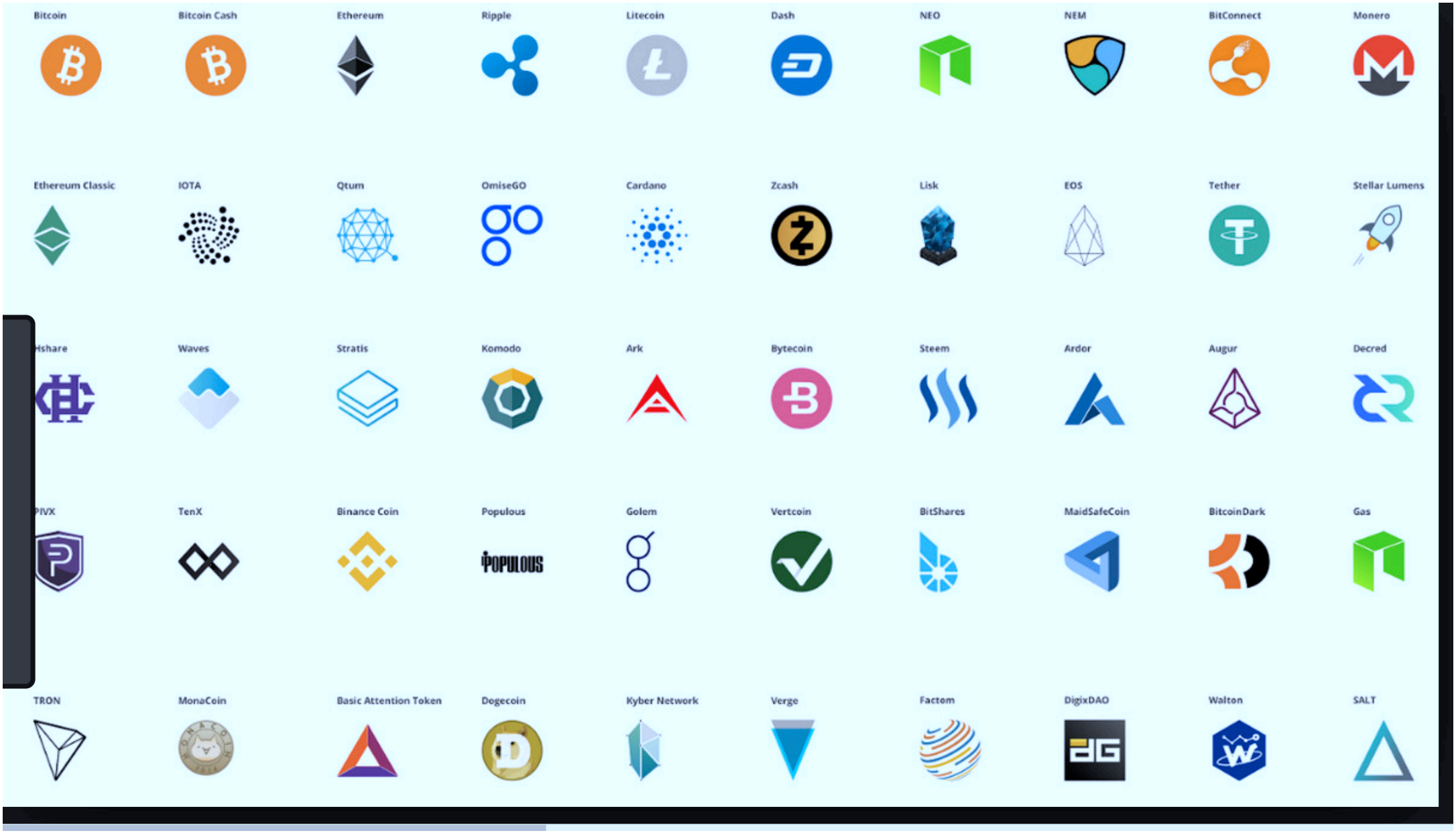
Relative trade volume (30 days)

Relative metrics use a log scale.

github.com/nickwb/cryptographic

Data sourced from coinmarketcap.com


The Blockchain Spectrum
Cryptography
Politics
Distributed Systems
Economics
Public/ Private Keypairs
Hash Functions
Merkle Trees
Error Correction
git + pgp
CAP Theorem
Append-only logs
BitTorrent
Horizontal Scalability
Byzantine Generals
Provably Secure Elections
Representation
Elections
Invisible Hand
Blockchains
PoW PoS
Game Theory
Espionage
Sovereignty
Democracy
Communities
Regulation
Markets
Intrinsic Value
Supply & Demand
Monetary Policy
Artificial Scarcity
National Security
Speculation
Global Banking
Resource Allocation
E-Commerce
@unchainedcap
unchained-capital.com


Bitcoin
Bitcoin Cash
Ethereum
Ripple
Litecoin
Dash
NEO
NEM
BitConnect
Monero
Ethereum Classic
IOTA
Qtum
OmiseGO
Cardano
Zcash
Lisk
EOS
Tether
Stellar Lumens
Hshare
Waves
Stratis
Komodo
Ark
Bytecoin
Steem
Ardor
Augur
Decred
PIVX
TenX
Binance Coin
Populous
POPULOUS
Golem
Vertcoin
BitShares
MaidSafeCoin
BitcoinDark
Gas
TRON
MonaCoin
Basic Attention Token
Dogecoin
Kyber Network
Verge
Factom
DigixDAO
Walton
SALT

coinmarketcap.com

#	Name	Market Cap	Price	Available Supply	Volume (24h)	% Change (24h)	Price Graph (7d)
1	Bitcoin	$ 10,419,201,130	$ 660.71	15,769,704 BTC	$ 95,166,700	-1.68 %	
2	Ethereum	$ 1,016,814,756	$ 12.37	82,186,773 ETH	$ 60,353,000	6.28 %	
3	Steem	$ 337,533,807	$ 3.75	90,122,449 STEEM	$ 1,458,810	-17.05 %	
4	Ripple	$ 226,059,937	$ 0.006379	35,438,257,609 XRP *	$ 791,535	-0.50 %	
5	Litecoin	$ 192,380,458	$ 4.11	46,751,704 LTC	$ 1,367,860	-0.96 %	
6	The DAO	$ 142,066,170	$ 0.123125	1,153,836,913 DAO *	$ 2,820,800	8.56 %	
7	NEM	$ 63,596,340	$ 0.007066	8,999,999,999 XEM *	$ 399,207	2.84 %	
8	Dash	$ 55,455,174	$ 8.39	6,606,352 DASH	$ 296,566	-1.48 %	
9	MaidSafeCoin	$ 32,524,218	$ 0.071868	452,552,412 MAID *	$ 129,285	-2.52 %	
10	Lisk	$ 30,851,200	$ 0.308512	100,000,000 LSK *	$ 452,353	-1.70 %	

BITCOIN VS ELECTRONIC MONEY

DIGITAL BUT DIFFERENT

Aside from being digital, there are few similarities between Bitcoin and e-money. Like credit cards, debit cards and Paypal, e-money is a mechanism to interact with government-issued and regulated currencies such as dollars and euros. In contrast, Bitcoin is an independent virtual currency that has no central regulation, can be transferred anonymously and may pose risks to consumers due to its market volatility.

BITCOIN AND E-MONEY: HOW THEY WORK

1 Transaction begins

BITCOIN

A user converts fiat currency for Bitcoins through an online exchange and stores them in a Bitcoin wallet.

E-MONEY

A user exchanges e-money for an equal amount of e-value stored, for example, in a mobile wallet.

2 Submitting payments

To make a purchase, a request is sent to Bitcoin's peer "miner" network using private encryption keys where the buy and sell order is validated.

The value of the money to be transferred and the recipient's phone number are entered. The money transfer is sent.

3 Verifying transactions

Bitcoin's decentralized peer-to-peer network of "miners" maintains a master ledger — a block chain — to verify every transaction. It takes an average of 10 minutes to verify. Upon verification, data is broadcast to all users and the ledger is updated.

User receives an SMS message to verify the mobile transaction. The e-money issuer maintains records of all transactions and customer balances.

BITCOIN IN BRIEF

What is it?
A peer-to-peer virtual currency without any fiat currency counterpart.

Who oversees it?
There is no central authority. Every transaction is authenticated by a collective network of Bitcoin users' computers.

How much is it worth?
Value fluctuates depending on demand and trust.

Is it anonymous?
It is mostly anonymous. Users create transaction "addresses" that are public but do not easily identify their owners.

How many will be in circulation?
Bitcoins will be created until 21 million are in circulation.

When did it start?
It was launched in 2009.

Who invented it?
That's a mystery. Satoshi Nakamoto is the creator's pseudonym.

Who uses Bitcoin and who accepts it?
Bitcoin seems to be used by people who easily have access to the Internet and formal financial services in general. A limited number of businesses accept Bitcoins as payment for goods or services.

KENYA'S E-MONEY SUCCESS STORY

M-PESA is a mobile phone-based money transfer system that was launched in Kenya in 2007.

Today, **93%** of Kenya's adult population are registered users.

60% (more than 11.6 million people) actively use M-PESA for person-to-person money transfers, bill paying and other transactions.

Mobile money is increasingly used across sub-Saharan Africa and other regions.

M-PESA

DIFFERENCES DEFINED

	Bitcoin	e-money
Accessibility:	*Largely limited to Internet connection*	*Access to electronic devices such as mobile phones, and an agent network*
Value:	*Determined by supply and demand, and trust in the system*	*Equal to amount of fiat currency exchanged into electronic form*
Customer ID:	*Anonymous*	*Financial Action Task Force standards apply for customer identification (though such standards permit simplified measures for lower risk financial products)*
Production:	*Mathematically generated ("mined") by peer network*	*Digitally issued against receipt of equal value of fiat currency of central authority*
Issuer:	*Community of developers, called "miners"*	*Legally established e-money issuer*
Regulator or oversight:	*None, though regulators are currently exploring*	*Regulated by central authority, typically central bank*

CGAP Advancing financial inclusion to improve the lives of the poor

Credit: SwitchYard Media

Digital Money Circle of Value

Quintessential Elements of a Viable Digital Currency

by Steven Dryall steve@incipient.ca

USER INTERFACE

Front-facing, direct user interaction

DOCUMENTATION

Open availability of formal and structured information

NETWORK

Supporting infrastructure and components

AWARENESS

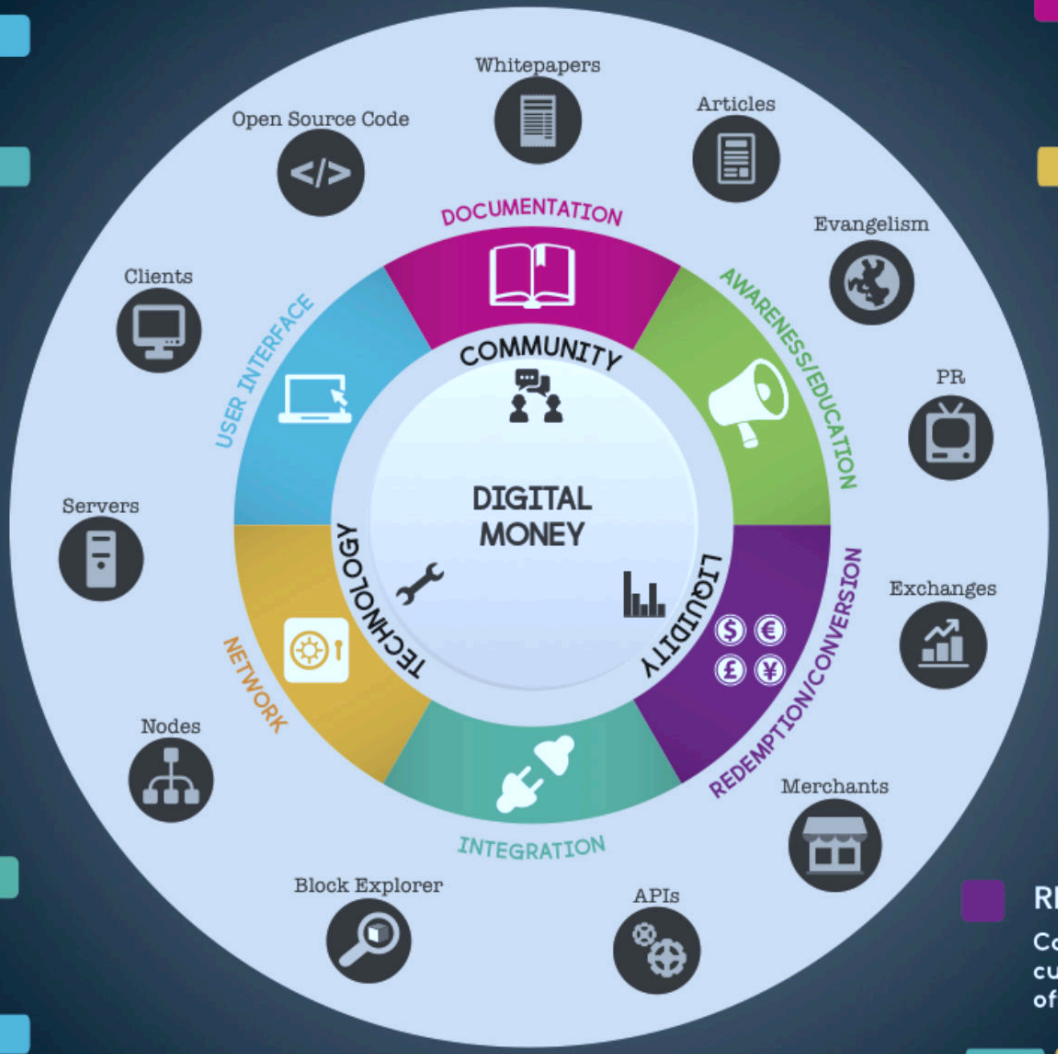
Education, information distribution and community building

INTEGRATION

Systems and APIS for integration with external systems and services

REDEMPTION/CONVERSION

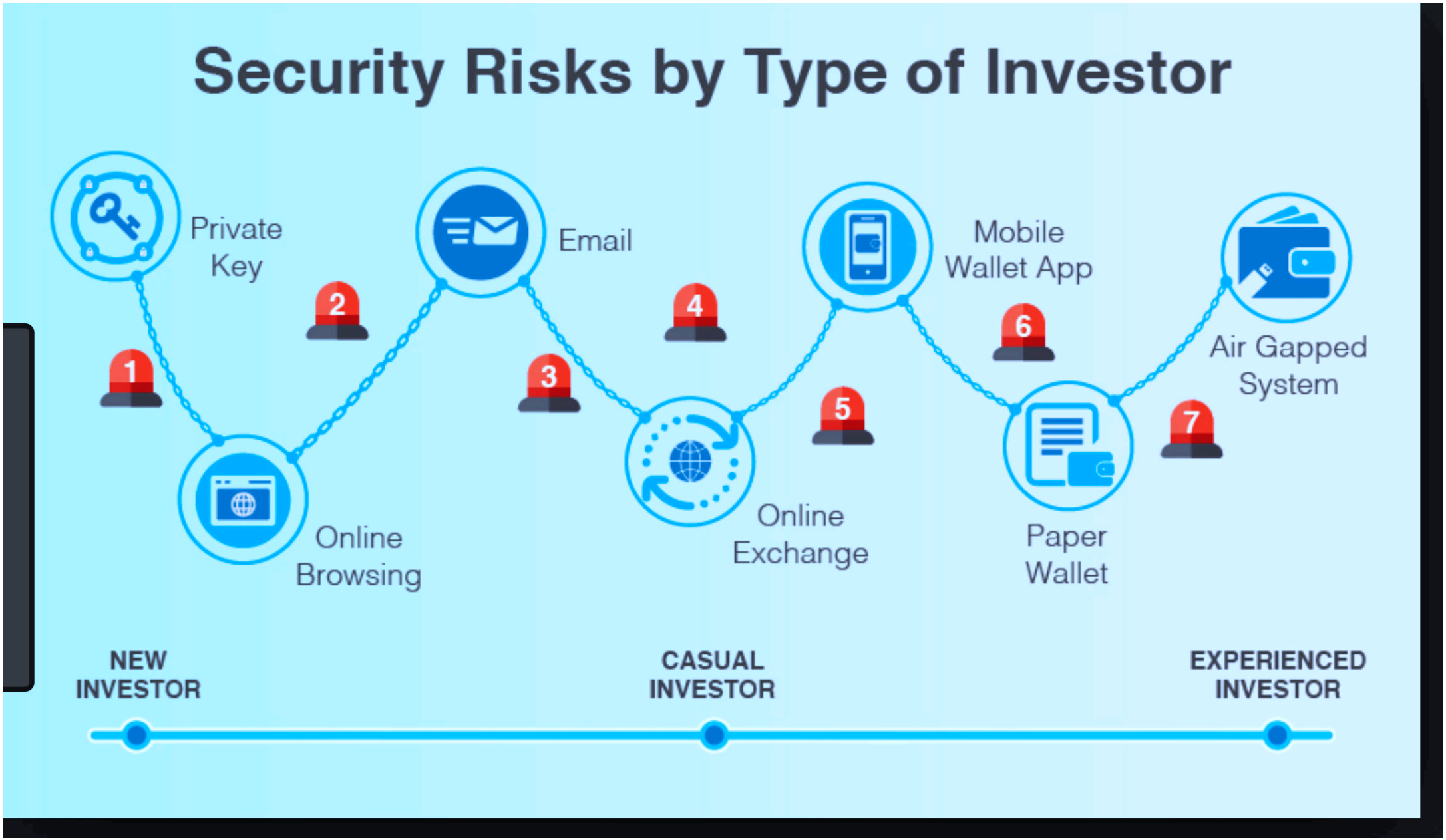
Conversion to other currencies or merchant offerings

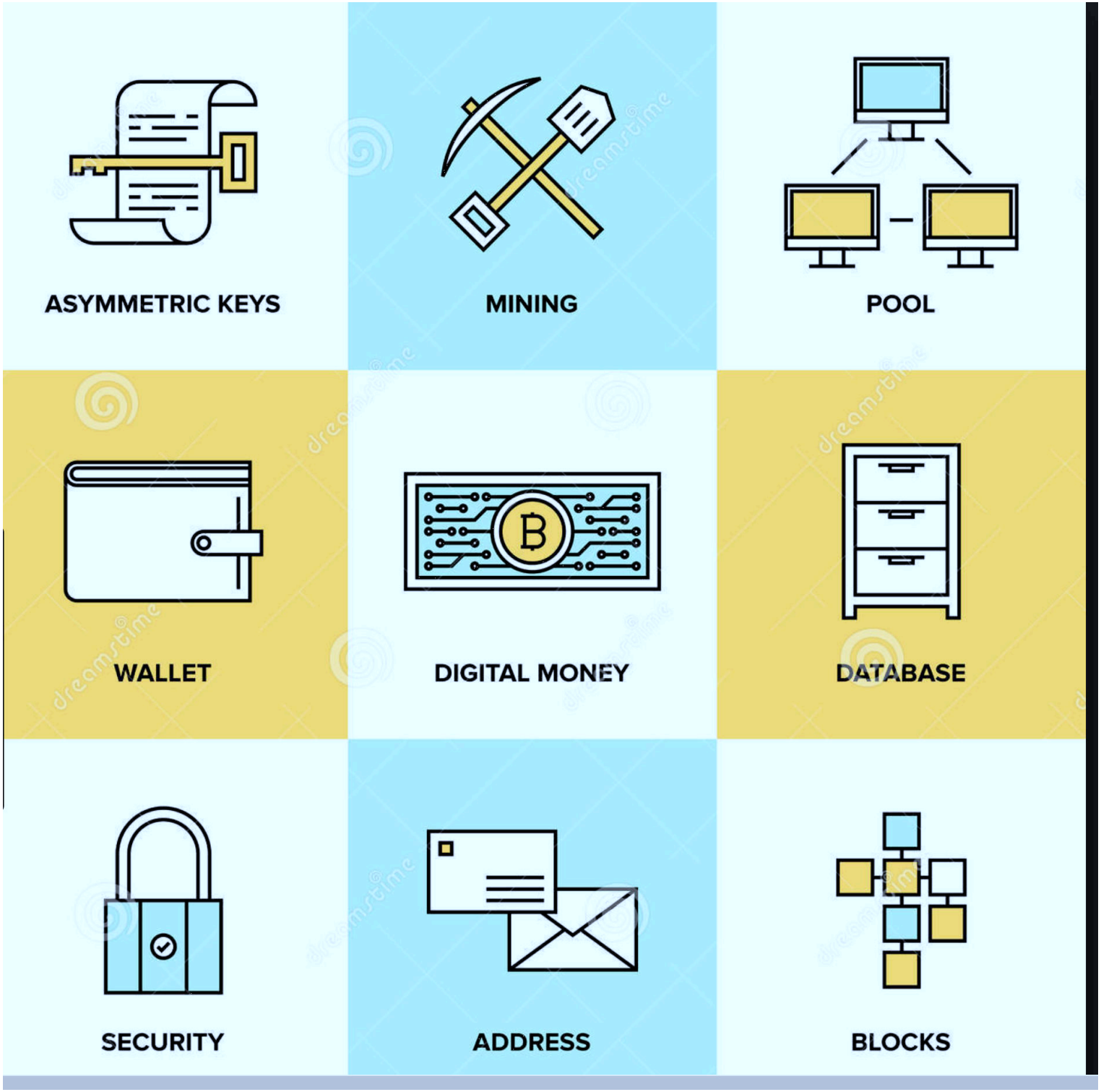


Incipient INDUSTRIES http://incipient.ca

Referral Foundation

ReferralCoins


Security Risks by Type of Investor
Private Key
Email
Mobile Wallet App
Air Gapped System
1
2
3
4
5
6
7
Online Browsing
Online Exchange
Paper Wallet
NEW INVESTOR
CASUAL INVESTOR
EXPERIENCED INVESTOR


ASYMMETRIC KEYS
MINING
POOL
WALLET
DIGITAL MONEY
DATABASE
SECURITY
ADDRESS
BLOCKS

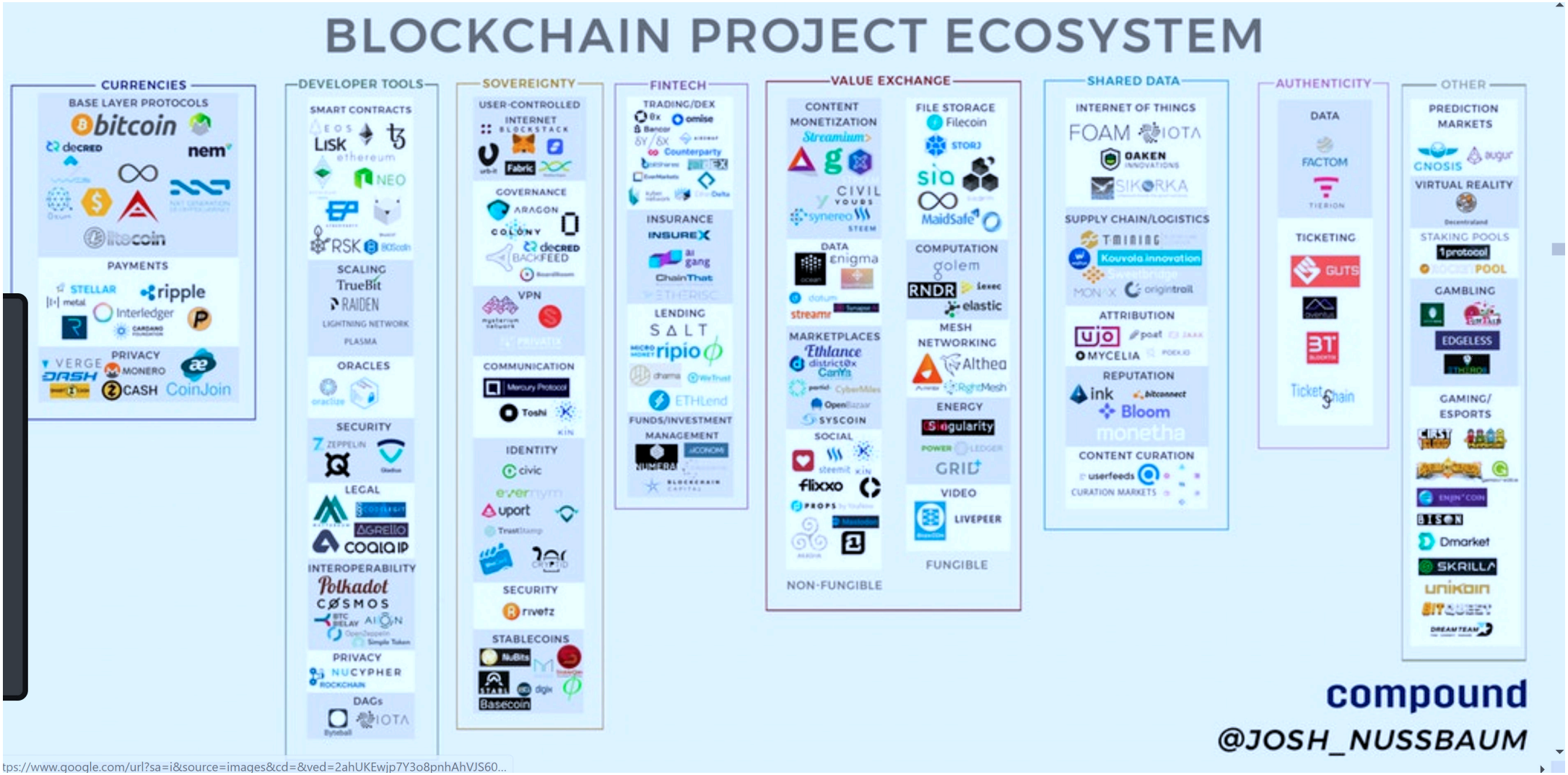


Figure 42: European and North American wallet users seem to prefer using local wallets

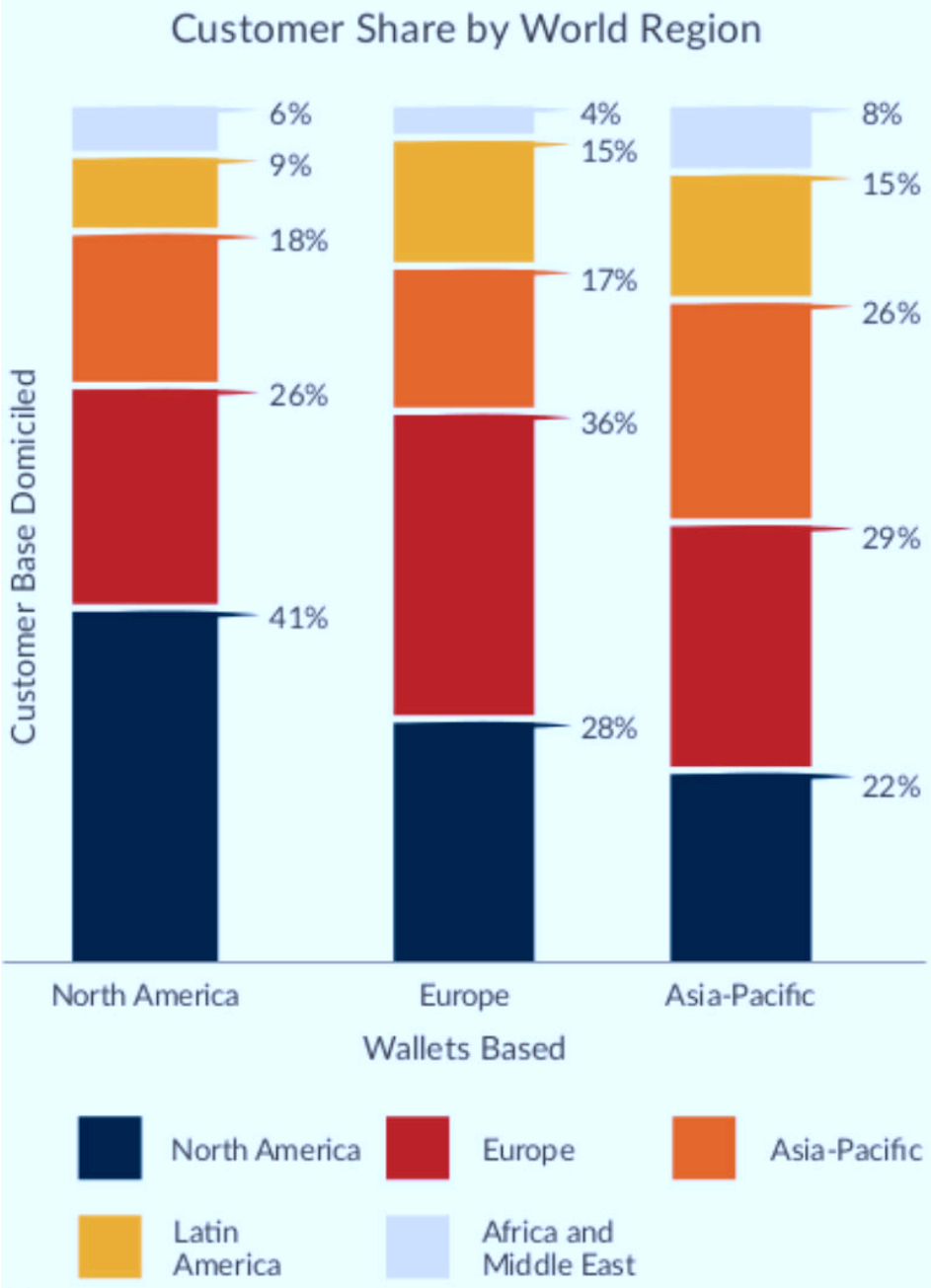
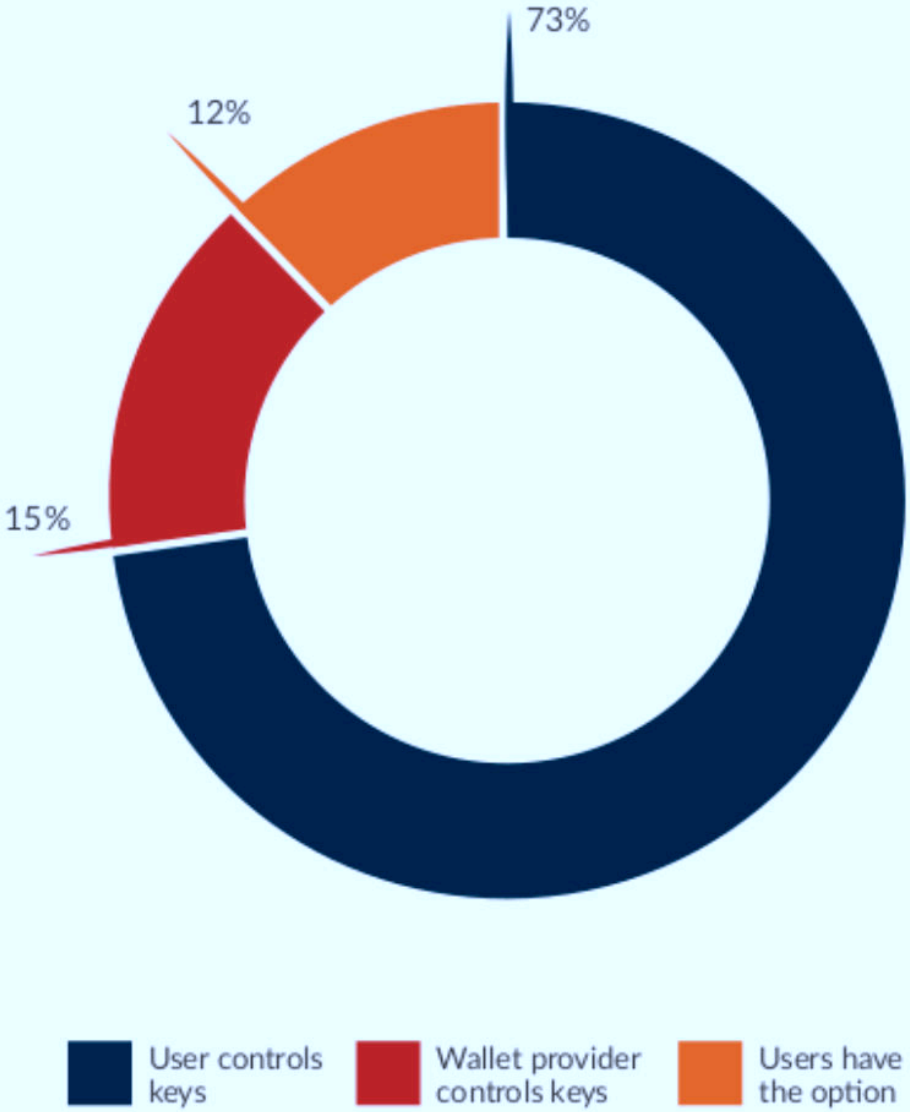
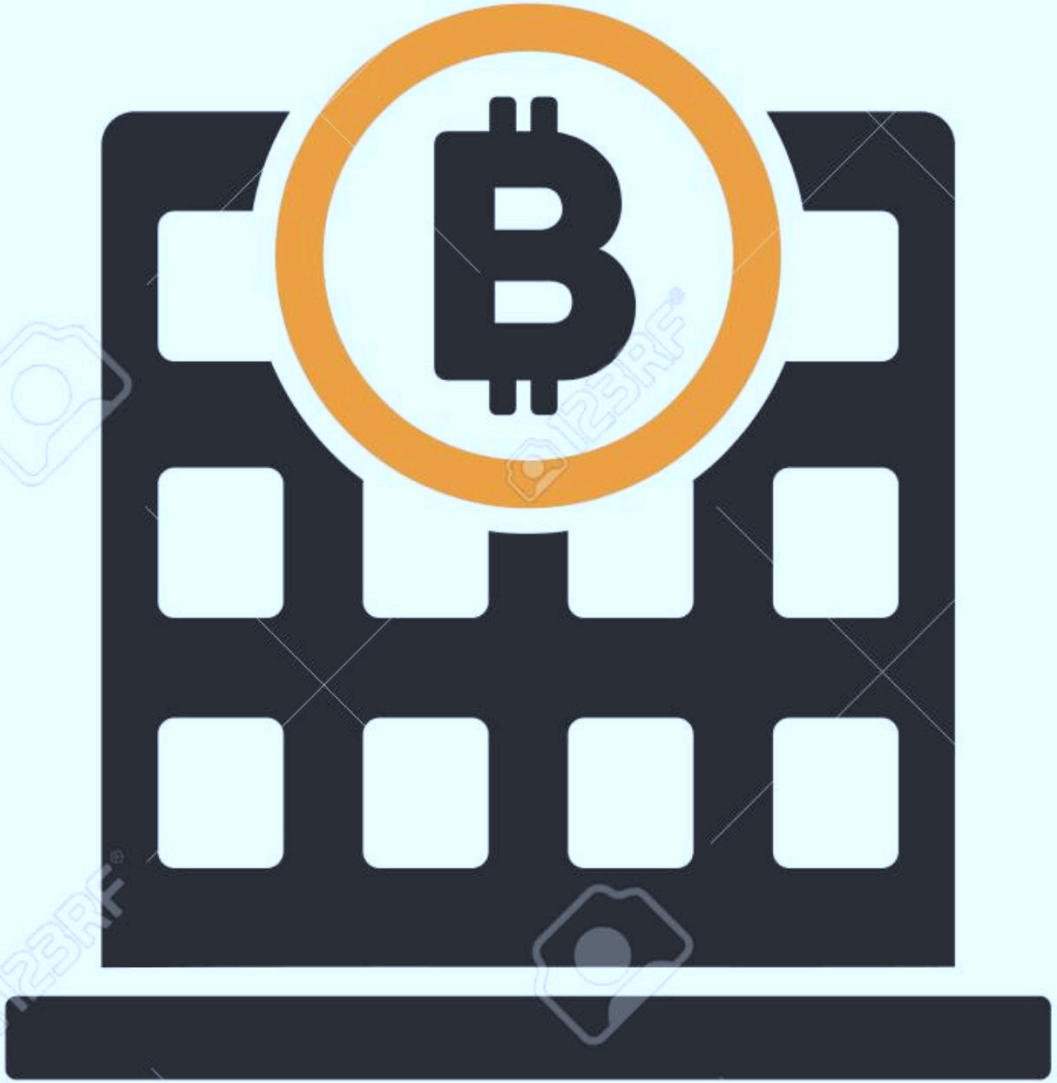


Figure 43: Over 70% of wallet providers do not control user funds



CRYPTOCURRENCY

300 ICONS



From Rich Dad's CashFlow Quadrant by Robert T. Kiyosaki

Active, Linear Income
Trade time for money
Money grows linearly
Work stops: income stops
Disabled: less income
Creating money to pay bills
Time is not your own

Passive, Residual Income
Leverage your time
Money grows exponentially
Work stops: income continues
Disabled: income continues
Creating true wealth
Total time freedom

4 Ways to Produce Income

5% Wealth

Employee

Have a J.O.B.

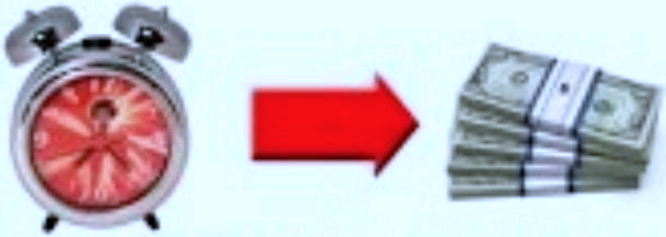
95% Wealth

Business Owner

Own a System

Self-Employed

Own a J.O.B.



95% Population

Investor

Own Investments

5% Population

Photocopy and share

CASH FLOW QUADRANTS

Photocopy and share

by best-selling author of "Rich Dad, Poor Dad" Robert T. Kiyosaki

Renting Your Life
Exchanging time for $
('Carrying buckets')

Owning Your Life
Owning non-managable income generating assets
('Building pipelines')

ACTIVE INCOME

PASSIVE INCOME

90% of the people
10% of the wealth

Way through quadrants

10% of the people
90% of the wealth

Way through quadrants

LEFT SIDE (Scarcity mentality)

RIGHT SIDE (Abundance mentality)

LIMITATIONS TO CHANGING SIDES:
Mind Set / Capital / Time / Education / Skills / Support

MIND SET DIFFERENCES

by best-selling author of "Secrets of the Millionaire Mind" T. Harv Eker

POOR AND MIDDLE CLASS

They believe "Life happens to them", victimized
They play the money game not to lose
They want to be rich
They think small, think limitations, wish and hope a lot
They focus on obstacles
They resent rich and successful people
They associate with negative and unsuccessful people
They think negatively about selling and promoting
They are smaller than their problems
They are poor receivers
They choose to get paid based on time
They think "either/or"
They focus on their working income - instant gratification
They mismanage their money, buy liabilities
They work hard for their money
They let fear stop them
They think they already know it all (HAVE, DO, BE)

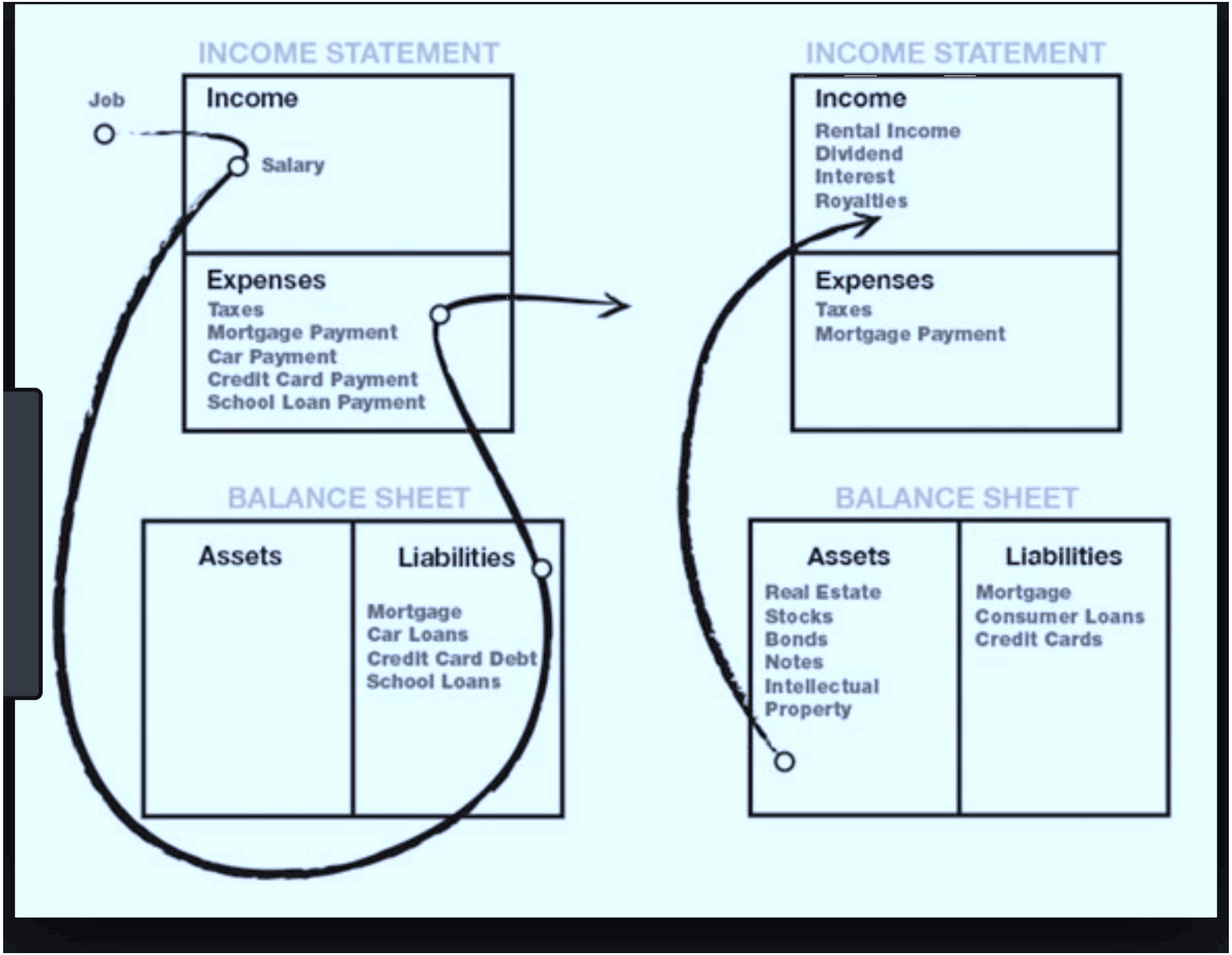
FINANCIALLY INDEPENDENT AND RICH

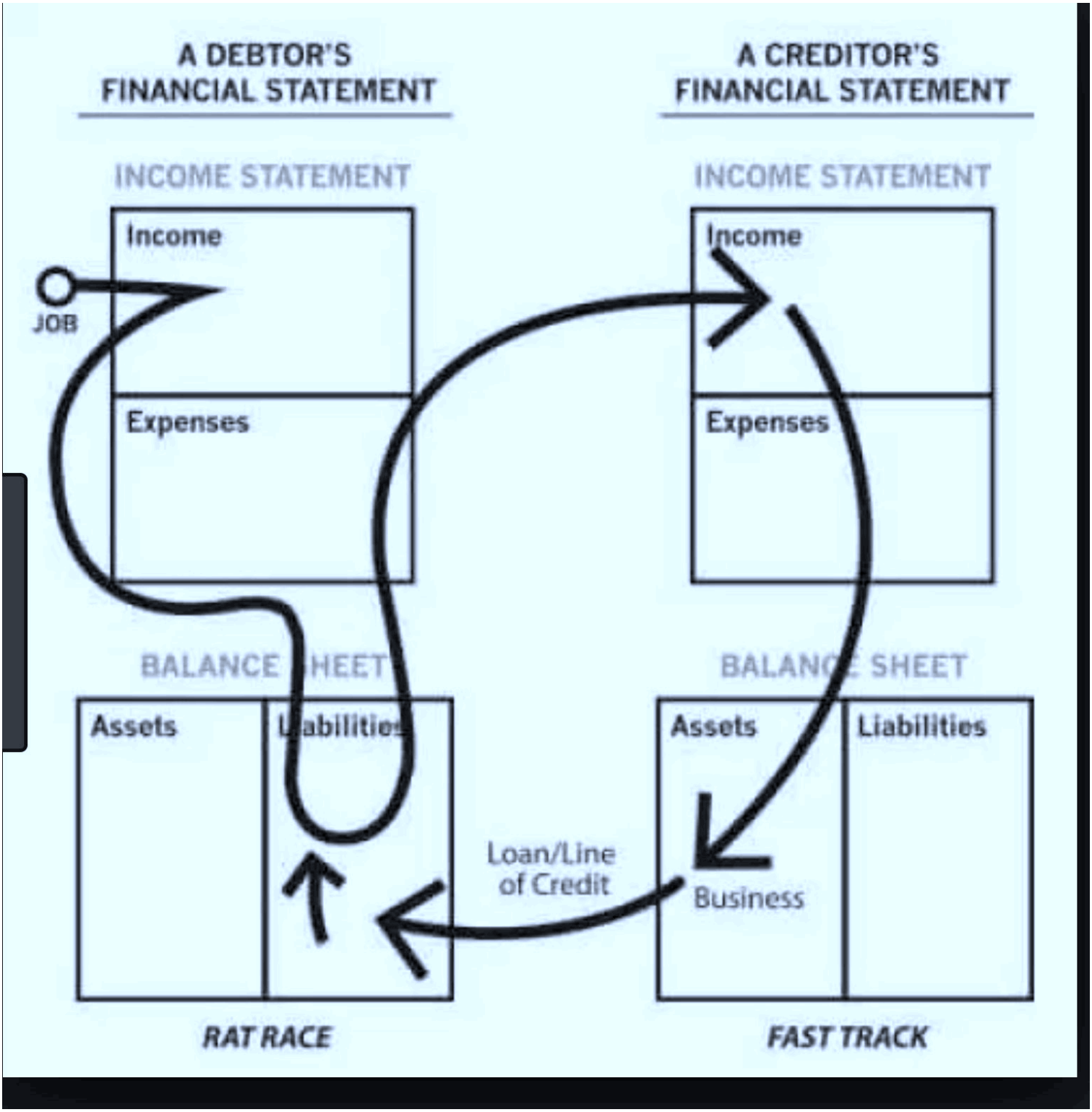
They believe "I create my life", accept full responsibility
They play the money game to win
They are committed to being rich
They think big, think possibilities, dream a lot
They focus on opportunities
They admire other rich and successful people
They associate with positive, successful people
They are willing to promote themselves and their values
They are bigger than their problems
They are excellent receivers
They choose to get paid based on results
They think "both"
They focus on their net worth - postponed gratification
They manage their money well, buy assets
They have their money work hard for them
They act in spite of fear
They constantly learn and grow (BE, DO, HAVE)

IF YOU WANT TO CHANGE YOUR LIFE, YOU HAVE TO CHANGE YOUR THINKING. PERIOD.

It is the hardest thing to change and it can be done. Decide, learn how and be patient with yourself.


LINEAR INCOME vs RESIDUAL INCOME
"I would rather earn 1% off a 100 people's efforts than 100% of my own efforts." ~ J. Paul Getty
100% x 1 = INCOME
YOU x JOB = INCOME
1% x 100 = INCOME
YOU x PEOPLE = INCOME
YOU HAVE A JOB
TIME = $
NO LEVERAGE
EMPLOYEE
BUSINESS OWNER
E
B
S
I
SELF EMPLOYED
INVESTOR
YOU OWN A SYSTEM & PEOPLE WORKS FOR YOU
PEOPLE = $$$
LEVERAGE
YOU OWN A JOB
TIME = $$
NO LEVERAGE
MONEY WORKS FOR YOU
$$$ = $$$$$
PASSIVE INCOME
Trading Time For Money
Starting Over Everyday at Zero
Income not Dependent
On your Presence
"the main reason why struggle financially is because they have spent years in school but learned nothing about money. The result is that people learn to work for money... but never learn to have money work for them." ~Robert Kiyosaki


INCOME STATEMENT
Job
Income
Salary
Expenses
Taxes
Mortgage Payment
Car Payment
Credit Card Payment
School Loan Payment
BALANCE SHEET
Assets
Liabilities
Mortgage
Car Loans
Credit Card Debt
School Loans
INCOME STATEMENT
Income
Rental Income
Dividend
Interest
Royalties
Expenses
Taxes
Mortgage Payment
BALANCE SHEET
Assets
Real Estate
Stocks
Bonds
Notes
Intellectual Property
Liabilities
Mortgage
Consumer Loans
Credit Cards


A DEBTOR'S FINANCIAL STATEMENT
INCOME STATEMENT
Income
Expenses
JOB
BALANCE SHEET
Assets
Liabilities
RAT RACE
A CREDITOR'S FINANCIAL STATEMENT
INCOME STATEMENT
Income
Expenses
BALANCE SHEET
Assets
Liabilities
Business
Loan/Line of Credit
FAST TRACK

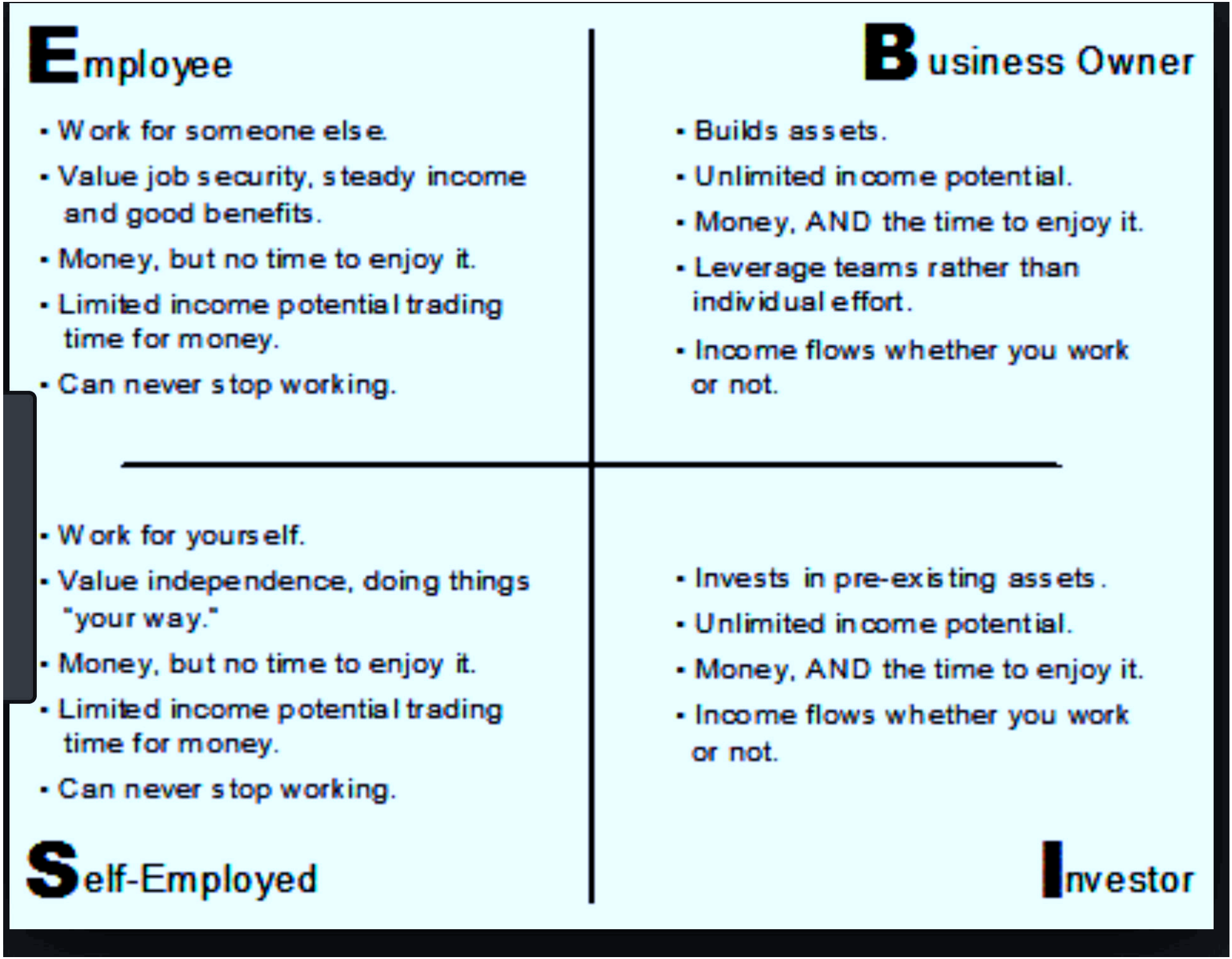
One of these Quadrants defines where you are

Employee
- Exchanges Time for Money T=$
- Income = 100% of one persons effort

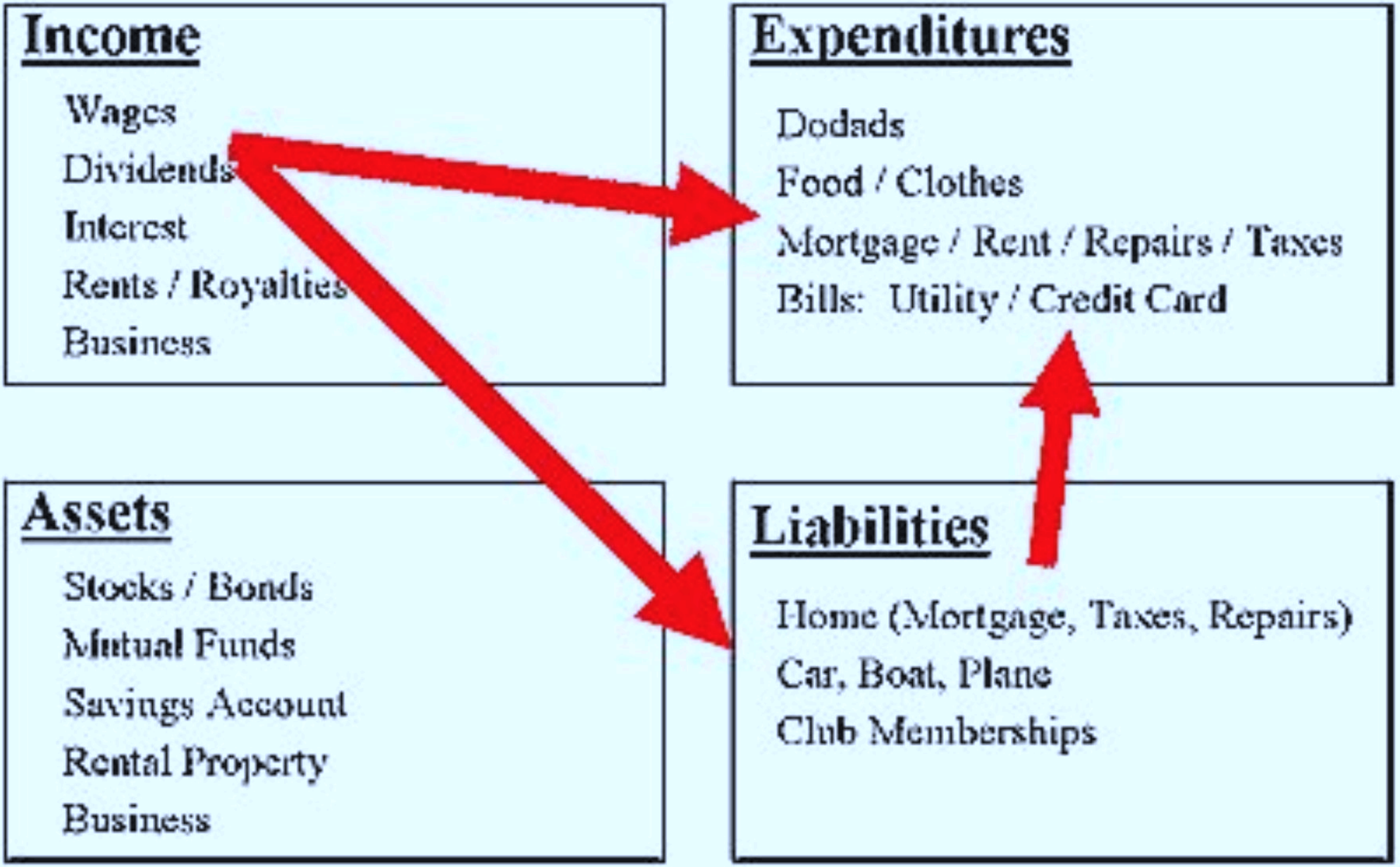
Self Employed
- Exchanges Time for Money T=$$
- You own a Job!

Business Owner
- Exchanges People for Income P=$$
- Has over 500 team members

Investor
- Exchanges Money for Money $$ =$$
- Has over ½ million Dollars invested


Employee
• Work for someone else.
• Value job security, steady income and good benefits.
• Money, but no time to enjoy it.
• Limited income potential trading time for money.
• Can never stop working.
Business Owner
• Builds assets.
• Unlimited income potential.
• Money, AND the time to enjoy it.
• Leverage teams rather than individual effort.
• Income flows whether you work or not.
• Work for yourself.
• Value independence, doing things "your way."
• Money, but no time to enjoy it.
• Limited income potential trading time for money.
• Can never stop working.
Self-Employed
• Invests in pre-existing assets.
• Unlimited income potential.
• Money, AND the time to enjoy it.
• Income flows whether you work or not.
Investor

What the Middle Class Buy on Payday (*Rich Dad Poor Dad*)



Python features

Lutz, *Programming Python*

no compiling or linking	rapid development cycle
no type declarations	simpler, shorter, more flexible
automatic memory management	garbage collection
high-level data types and operations	fast development
object-oriented programming	code structuring and reuse, C++
embedding and extending in C	mixed language systems
classes, modules, exceptions	"programming-in-the-large" support
dynamic loading of C modules	simplified extensions, smaller binaries
dynamic reloading of C modules	programs can be modified without stopping

Hello World

Let's get stuck in, and what better way than with the programmer's best friend, the 'Hello World' application! Start by opening a terminal. Its current working directory will be your home directory. It's probably a good idea to make a directory for the files we'll be creating in this tutorial, rather than having them loose in your home directory. You can create a directory called Python using the command `mkdir Python`. You'll then want to change into that directory using the command `cd Python`.

The next step is to create an empty file using the command 'touch' followed by the filename. Our expert used the command `touch hello_world.py`. The final and most important part of setting up the file is making it executable. This allows us to run code inside the hello_world.py file. We do this with the command `chmod +x hello_world.py`. Now that we have our file set up, we can go ahead and open it up in nano, or any text editor of your choice. Gedit is a great editor with syntax highlighting support that should be available on any distribution. You'll be able to install it using your package manager if you don't have it already.

```
[liam@liam-laptop ~]$ mkdir Python
[liam@liam-laptop ~]$ cd Python/
[liam@liam-laptop Python]$ touch hello_world.py
[liam@liam-laptop Python]$ chmod +x hello_world.py
[liam@liam-laptop Python]$ nano hello_world.py
```

Our Hello World program is very simple, it only needs two lines. The first line begins with a 'shebang' (the symbol #! – also known as a hashbang) followed by the path to the Python interpreter. The program loader uses this line to work out what the rest of the lines need to be interpreted with. If you're running this in an IDE like IDLE, you don't necessarily need to do this.

The code that is actually read by the Python interpreter is only a single line. We're passing the value Hello World to the print function by placing it in brackets immediately after we've called the print function. Hello World is enclosed in quotation marks to indicate that it is a literal value and should not be interpreted as source code. As expected, the print function in Python prints any value that gets passed to it from the console.

You can save the changes you've just made to the file in nano using the key combination `Ctrl+O`, followed by Enter. Use `Ctrl+X` to exit nano.

```
#!/usr/bin/env python2
print("Hello World")
```

You can run the Hello World program by prefixing its filename with `./` – in this case you'd type: `./hello_world.py`.

```
[liam@liam-laptop Python]$ ./hello_world.py
Hello World
```

TIP

If you were using a graphical editor such as gedit, then you would only have to do the last step of making the file executable. You should only have to mark the file as executable once. You can freely edit the file once it is executable.

Variables and data types

A variable is a name in source code that is associated with an area in memory that you can use to store data, which is then called upon throughout the code. The data can be one of many types, including:

Integer	Stores whole numbers
Float	Stores decimal numbers
Boolean	Can have a value of True or False
String	Stores a collection of characters. "Hello World" is a string

As well as these main data types, there are sequence types (technically, a string is a sequence type but is so commonly used we've classed it as a main data type):

List	Contains a collection of data in a specific order
Tuple	Contains a collection immutable data in a specific order

A tuple would be used for something like a co-ordinate, containing an x and y value stored as a single variable, whereas a list is typically used to store larger collections. The data stored in a tuple is immutable because you aren't able to change values of individual elements in a tuple. However, you can do so in a list.

It will also be useful to know about Python's dictionary type. A dictionary is a mapped data type. It stores data in key-value pairs. This means that you access values stored in the dictionary using that value's corresponding key, which is different to how you would do it with a list. In a list, you would access an element of the list using that element's index (a number representing the element's position in the list).

Let's work on a program we can use to demonstrate how to use variables and different data types. It's worth noting at this point that you don't always have to specify data types in Python. Feel free to create this file in any editor you like. Everything will work just fine as long as you remember to make the file executable. We're going to call ours `variables.py`.

> "A variable is a name in source code that is associated with an area in memory that you can use to store data"

Interpreted vs compiled languages

An interpreted language such as Python is one where the source code is converted to machine code and then executed each time the program runs. This is different from a compiled language such as C, where the source code is only converted to machine code once – the resulting machine code is then executed each time the program runs.

```
#!/usr/bin/env python2

# We create a variable by writing the name of the variable we want followed
# by an equals sign, which is followed by the value we want to store in the
# variable. For example, the following line creates a variable called
# hello_str, containing the string Hello World.
hello_str = "Hello World"
```

The following line creates an integer variable called hello_int with the # value of 21. Notice how it doesn't need to go in quotation marks

```
hello_int = 21
```

The same principal is true of Boolean values

```
hello_bool = True
```

We create a tuple in the following way

```
hello_tuple = (21, 32)
```

And a list in this way

```
hello_list = ["Hello,","this","is","a","list"]

# This list now contains 5 strings. Notice that there are no spaces
# between these strings so if you were to join them up so make a sentence
# you'd have to add a space between each element.
```

You could also create the same list in the following way

```
hello_list = list()
hello_list.append("Hello,")
hello_list.append("this")
hello_list.append("is")
hello_list.append("a")
hello_list.append("list")

# The first line creates an empty list and the following lines use the append
# function of the list type to add elements to the list. This way of using a
# list isn't really very useful when working with strings you know of in
# advance, but it can be useful when working with dynamic data such as user
# input. This list will overwrite the first list without any warning as we
# are using the same variable name as the previous list.
```

We might as well create a dictionary while we're at it. Notice how we've aligned the colons below to make the code tidy

```
hello_dict = { "first_name" : "Liam",
               "last_name"  : "Fraser",
               "eye_colour" : "Blue" }

# Let's access some elements inside our collections
# We'll start by changing the value of the last string in our hello_list and
# add an exclamation mark to the end. The "list" string is the 5th element
# in the list. However, indexes in Python are zero-based, which means the
# first element has an index of 0.
```

Notice that there will now be two exclamation marks when we print the element

```
print(hello_list[4])
hello_list[4] += "!"
# The above line is the same as
hello_list[4] = hello_list[4] + "!"
print(hello_list[4])
```

TIP

At this point, it's worth explaining that any text in a Python file that follows a # character will be ignored by the interpreter. This is so you can write comments in your code.

"Any text in a Python file that follows a # character will be ignored"

Remember that tuples are immutable, although we can access the elements of them like so

Let's create a sentence using the data in our hello_dict

A tidier way of doing this would be to use Python's string formatter

```
print(str(hello_tuple[0]))
# We can't change the value of those elements like we just did with the list
# Notice the use of the str function above to explicitly convert the integer
# value inside the tuple to a string before printing it.

print(hello_dict["first_name"] + " " + hello_dict["last_name"] + " has " +
    hello_dict["eye_colour"] + " eyes.")

print("{0} {1} has {2} eyes.".format(hello_dict["first_name"],
                                     hello_dict["last_name"],
                                     hello_dict["eye_colour"]))
```

Control structures

In programming, a control structure is any kind of statement that can change the path that the code execution takes. For example, a control structure that decided to end the program if a number was less than 5 would look something like this:

```
#!/usr/bin/env python2

import sys # Used for the sys.exit function

int_condition = 5

if int_condition < 6:
    sys.exit("int_condition must be >= 6")
else:
    print("int_condition was >= 6 - continuing")
```

The path that the code takes will depend on the value of the integer int_condition. The code in the 'if' block will only be executed if the condition is true. The import statement is used to load the Python system library; the latter provides the exit function, allowing you to exit the program, printing an error message. Notice that indentation (in this case four spaces per indent) is used to indicate which statement a block of code belongs to.

'If' statements are probably the most commonly used control structures. Other control structures include:

- For statements, which allow you to iterate over items in collections, or to repeat a piece of code a certain number of times;
- While statements, a loop that continues while the condition is true.

We're going to write a program that accepts user input from the user to demonstrate how control structures work. We're calling it construct.py.

The 'for' loop is using a local copy of the current value, which means any changes inside the loop won't make any changes affecting the list. On the other hand however, the 'while' loop is directly accessing elements in the list, so you could change the list there should you want to do so. We will talk about variable scope in some more detail later on. The output from the above program is as follows:

More about a Python list

A Python list is similar to an array in other languages. A list (or tuple) in Python can contain data of multiple types, which is not usually the case with arrays in other languages. For this reason, we recommend that you only store data of the same type in a list. This should almost always be the case anyway due to the nature of the way data in a list would be processed.

Indentation in detail

As previously mentioned, the level of indentation dictates which statement a block of code belongs to. Indentation is mandatory in Python, whereas in other languages, sets of braces are used to organise code blocks. For this reason, it is essential that you use a consistent indentation style. Four spaces are typically used to represent a single level of indentation in Python. You can use tabs, but tabs are not well defined, especially if you happen to open a file in more than one editor.

"The 'for' loop uses a local copy, so changes in the loop won't affect the list"

```
[liam@liam-laptop Python]$ ./construct.py
How many integers? acd
You must enter an integer

[liam@liam-laptop Python]$ ./construct.py
How many integers? 3
Please enter integer 1: t
You must enter an integer
Please enter integer 1: 5
Please enter integer 2: 2
Please enter integer 3: 6
Using a for loop
5
2
6
Using a while loop
5
2
6
```

The number of integers we want in the list

A list to store the integers

These are used to keep track of how many integers we currently have

If the above succeeds then isint will be set to true: isint =True

By now, the user has given up or we have a list filled with integers. We can loop through these in a couple of ways. The first is with a for loop

```
#!/usr/bin/env python2

# We're going to write a program that will ask the user to input an arbitrary
# number of integers, store them in a collection, and then demonstrate how the
# collection would be used with various control structures.

import sys # Used for the sys.exit function

target_int = raw_input("How many integers? ")

# By now, the variable target_int contains a string representation of
# whatever the user typed. We need to try and convert that to an integer but
# be ready to # deal with the error if it's not. Otherwise the program will
# crash.
try:
    target_int = int(target_int)
except ValueError:
    sys.exit("You must enter an integer")


ints = list()

count = 0

# Keep asking for an integer until we have the required number
while count < target_int:
    new_int = raw_input("Please enter integer {0}: ".format(count + 1))
    isint = False
    try:
        new_int = int(new_int)

    except:
        print("You must enter an integer")

    # Only carry on if we have an integer. If not, we'll loop again
    # Notice below I use ==, which is different from =. The single equals is an
    # assignment operator whereas the double equals is a comparison operator.

    if isint == True:
        # Add the integer to the collection
        ints.append(new_int)
        # Increment the count by 1
        count += 1


print("Using a for loop")
for value in ints:
    print(str(value))
```

TIP

You can define defaults for variables if you want to be able to call the function without passing any variables through at all. You do this by putting an equals sign after the variable name. For example, you can do: `def modify_string (original=" Default String")`

```
# Or with a while loop:
print("Using a while loop")
# We already have the total above, but knowing the len function is very
# useful.
total = len(ints)
count = 0
while count < total:
    print(str(ints[count]))
    count += 1
```

Functions and variable scope

Functions are used in programming to break processes down into smaller chunks. This often makes code much easier to read. Functions can also be reusable if designed in a certain way. Functions can have variables passed to them. Variables in Python are always passed by value, which means that a copy of the variable is passed to the function that is only valid in the scope of the function. Any changes made to the original variable inside the function will be discarded. However, functions can also return values, so this isn't an issue. Functions are defined with the keyword def, followed by the name of the function. Any variables that can be passed through are put in brackets following the function's name. Multiple variables are separated by commas. The names given to the variables in these brackets are the ones that they will have in the scope of the function, regardless of what the variable that's passed to the function is called. Let's see this in action.

The output from the program opposite is as follows:

> "Functions are used in programming to break processes down in"

```
#!/usr/bin/env python2

# Below is a function called modify_string, which accepts a variable
# that will be called original in the scope of the function. Anything
# indented with 4 spaces under the function definition is in the
# scope.
def modify_string(original):
    original += " that has been modified."
    # At the moment, only the local copy of this string has been modified

def modify_string_return(original):
    original += " that has been modified."
    # However, we can return our local copy to the caller. The function
    # ends as soon as the return statement is used, regardless of where it
    # is in the function.
    return original


test_string = "This is a test string"

modify_string(test_string)
print(test_string)

test_string = modify_string_return(test_string)
print(test_string)

# The function's return value is stored in the variable test string,
# overwriting the original and therefore changing the value that is
# printed.
```

We are now outside of the scope of the modify_string function, as we have reduced the level of indentation

The test string won't be changed in this code

However, we can call the function like this

```
[liam@liam-laptop Python]$ ./functions_and_scope.py
This is a test string
This is a test string that has been modified.
```

Scope is an important thing to get the hang of, otherwise it can get you into some bad habits. Let's write a quick program to demonstrate this. It's going to have a Boolean variable called cont, which will decide if a number will be assigned to a variable in an if statement. However, the variable hasn't been defined anywhere apart from in the scope of the if statement. We'll finish off by trying to print the variable.

```
#!/usr/bin/env python2
cont = False
if cont:
    var = 1234
print(var)
```

In the section of code above, Python will convert the integer to a string before printing it. However, it's always a good idea to explicitly convert things to strings – especially when it comes to concatenating strings together. If you try to use the + operator on a string and an integer, there will be an error because it's not explicitly clear what needs to happen. The + operator would usually add two integers together. Having said that, Python's string formatter that we demonstrated earlier is a cleaner way of doing that. Can you see the problem? Var has only been defined in the scope of the if statement. This means that we get a very nasty error when we try to access var.

```
[liam@liam-laptop Python]$ ./scope.py
Traceback (most recent call last):
  File "./scope.py", line 8, in <module>
    print var
NameError: name 'var' is not defined
```

If cont is set to True, then the variable will be created and we can access it just fine. However, this is a bad way to do things. The correct way is to initialise the variable outside of the scope of the if statement.

```
#!/usr/bin/env python2

cont = False

var = 0
if cont:
    var = 1234

if var != 0:
    print(var)
```

The variable var is defined in a wider scope than the if statement, and can still be accessed by the if statement. Any changes made to var inside the if statement are changing the variable defined in the larger scope. This example doesn't really do anything useful apart from illustrate the potential problem, but the worst-case scenario has gone from the program crashing to printing a zero. Even that doesn't happen because we've added an extra construct to test the value of var before printing it.

Coding style

It's worth taking a little time to talk about coding style. It's simple to write tidy code. The key is consistency. For example, you should always name your variables in the same manner. It doesn't matter if you want to use camelCase or use underscores as we have. One crucial thing is to use self-documenting identifiers for variables. You shouldn't have to guess what a variable does. The other thing that goes with this is to always comment your code. This will help anyone else who reads your code, and yourself in the future. It's also useful to put a brief summary at the top of a code file describing what the application does, or a part of the application if it's made up of multiple files.

Comparison operators

The common comparison operators available in Python include:

<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal

Summary

This article should have introduced you to the basics of programming in Python. Hopefully you are getting used to the syntax, indentation and general look and feel of a Python program. The next step is to learn how to come up with a problem that you want to solve, and break it down into small enough steps that you can implement in a programming language.

Google, or any other search engine, is very helpful. If you are stuck with anything, or have an error message you can't work out how to fix, stick it into Google and you should be a lot closer to solving your problem. For example, if we Google 'play mp3 file with python', the first link takes us to a Stack Overflow thread with a bunch of useful replies. Don't be afraid to get stuck in – the real fun of programming is solving problems one manageable chunk at a time.

Happy programming!

50 ESSENTIAL PYTHON COMMANDS

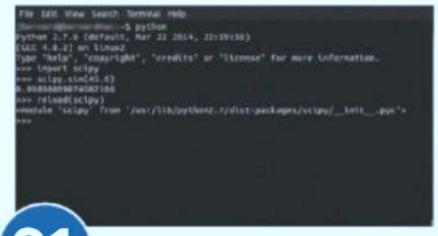
Python is known as a very dense language, with lots of modules capable of doing almost anything. Here, we will look at the core essentials that everyone needs to know

Python has a massive environment of extra modules that can provide functionality in hundreds of different disciplines. However, every programming language has a core set of functionality that everyone should know in order to get useful work done. Python is no different in this regard. Here, we will look at 50 commands that we consider to be essential to programming in Python. Others may pick a slightly different set, but this list contains the best of the best.

We will cover all of the basic commands, from importing extra modules at the beginning of a program to returning values to the calling environment at the end. We will also be looking at some commands that are useful in learning about the current session within Python, like the current list of variables that have been defined and how memory is being used.

Because the Python environment involves using a lot of extra modules, we will also look at a few commands that are strictly outside of Python. We will see how to install external modules and how to manage multiple environments for different development projects. Since this is going to be a list of commands, there is the assumption that you already know the basics of how to use loops and conditional structures. This piece is designed to help you remember commands that you know you've seen before, and hopefully introduce you to a few that you may not have seen yet.

Although we've done our best to pack everything you could ever need into 50 tips, Python is such an expansive language that some commands will have been left out. Make some time to learn about the ones that we didn't cover here, once you've mastered these.



01 Importing modules

The strength of Python is its ability to be extended through modules. The first step in many programs is to import those modules that you need. The simplest import statement is to just call 'import modulename'. In this case, those functions and objects provided are not in the general namespace. You need to call them using the complete name (modulename.methodname). You can shorten the 'modulename' part with the command 'import modulename as mn'. You can skip this issue completely with the command 'from modulename import *' to import everything from the given module. Then you can call those provided capabilities directly. If you only need a few of the provided items, you can import them selectively by replacing the '*' with the method or object names.

02 Reloading modules

When a module is first imported, any initialisation functions are run at that time. This may involve creating data objects, or initiating connections. But, this is only done the first time within a given session. Importing the same module again won't re-execute any of the initialisation code. If you want to have this code re-run, you need to use the reload command. The format is 'reload(modulename)'. Something to keep in mind is that the dictionary from the previous import isn't dumped, but only written over. This means that any definitions that have changed between the import and the reload are updated correctly. But if you delete a definition, the old one will stick around and still be accessible. There may be other side effects, so always use with caution.

03 Installing new modules

While most of the commands we are looking at are Python commands that are to be executed within a Python session, there are a few essential commands that need to be executed outside of Python. The first of these is pip. Installing a module involves downloading the source code, and compiling any included external code. Luckily, there is a repository of hundreds of Python modules available at http://pypi.python.org. Instead of doing everything manually, you can install a new module by using the command 'pip install modulename'. This command will also do a dependency check and install any missing modules before installing the one you requested. You may need administrator rights if you want this new module installed in the global library for your computer. On a Linux machine, you would simply run the pip command with sudo. Otherwise, you can install it to your personal library directory by adding the command line option '—user'.

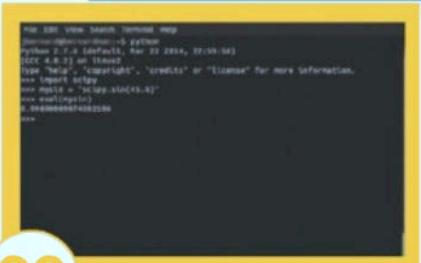
"Every programming language out there has a core set of functionality that everyone should know in order to get useful work done. Python is no different"

04 Executing a script

Importing a module does run the code within the module file, but does it through the module maintenance code within the Python engine. This maintenance code also deals with running initialising code. If you only wish to take a Python script and execute the raw code within the current session, you can use the 'execfile("filename.py")' command, where the main option is a string containing the Python file to load and execute. By default, any definitions are loaded into the locals and globals of the current session. You can optionally include two extra parameters the execfile command. These two options are both dictionaries, one for a different set of locals and a different set of globals. If you only hand in one dictionary, it is assumed to be a globals dictionary. The return value of this command is None.

05 An enhanced shell

The default interactive shell is provided through the command 'python', but is rather limited. An enhanced shell is provided by the command 'ipython'. It provides a lot of extra functionality to the code developer. A thorough history system is available, giving you access to not only commands from the current session, but also from previous sessions. There are also magic commands that provide enhanced ways of interacting with the current Python session. For more complex interactions, you can create and use macros. You can also easily peek into the memory of the Python session and decompile Python code. You can even create profiles that allow you to handle initialisation steps that you may need to do every time you use iPython.



06 Evaluating code

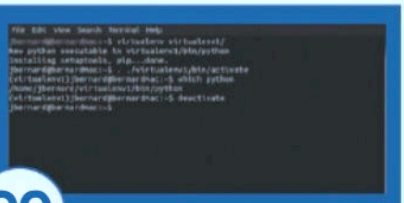
Sometimes, you may have chunks of code that are put together programmatically. If these pieces of code are put together as a string, you can execute the result with the command 'eval("code_string")'. Any syntax errors within the code string are reported as exceptions. By default, this code is executed within the current session, using the current globals and locals dictionaries. The 'eval' command can also take two other optional parameters, where you can provide a different set of dictionaries for the globals and locals. If there is only one additional parameter, then it is assumed to be a globals dictionary. You can optionally hand in a code object that is created with the compile command instead of the code string. The return value of this command is None.

07 Asserting values

At some point, we all need to debug some piece of code we are trying to write. One of the tools useful in this is the concept of an assertion. The assert command takes a Python expression and checks to see if it is true. If so, then execution continues as normal. If it is not true, then an AssertionError is raised. This way, you can check to make sure that invariants within your code stay invariant. By doing so, you can check assumptions made within your code. You can optionally include a second parameter to the assert command. This second parameter is Python expression that is executed if the assertion fails. Usually, this is some type of detailed error message that gets printed out. Or, you may want to include cleanup code that tries to recover from the failed assertion.

08 Mapping functions

A common task that is done in modern programs is to map a given computation to an entire list of elements. Python provides the command 'map()' to do just this. Map returns a list of the results of the function applied to each element of an iterable object. Map can actually take more than one function and more than one iterable object. If it is given more than one function, then a list of tuples is returned, with each element of the tuple containing the results from each function. If there is more than one iterable handed in, then map assumes that the functions take more than one input parameter, so it will take them from the given iterables. This has the implicit assumption that the iterables are all of the same size, and that they are all necessary as parameters for the given function.



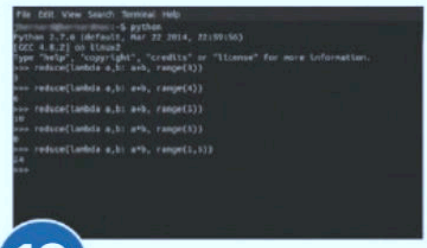
09 Virtualenvs

Because of the potential complexity of the Python environment, it is sometimes best to set up a clean environment within which to install only the modules you need for a given project. In this case, you can use the virtualenv command to initialise such an environment. If you create a directory named 'ENV', you can create a new environment with the command 'virtualenv ENV'. This will create the subdirectories bin, lib and include, and populate them with an initial environment. You can then start using this new environment by sourcing the script 'ENV/bin/activate', which will change several environment variables, such as the PATH. When you are done, you can source the script 'ENV/bin/deactivate' to reset your shell's environment back to its previous condition. In this way, you can have environments that only have the modules you need for a given set of tasks.

"While not strictly commands, everyone needs to know how to deal with loops. The two main types of loops are a fixed number of iterations loop (for) and a conditional loop (while)"

10 Loops

While not strictly commands, everyone needs to know how to deal with loops. The two main types of loops are a fixed number of iterations loop (for) and a conditional loop (while). In a for loop, you iterate over some sequence of values, pulling them off the list one at a time and putting them in a temporary variable. You continue until either you have processed every element or you have hit a break command. In a while loop, you continue going through the loop as long as some test expression evaluates to True. While loops can also be exited early by using the break command, you can also skip pieces of code within either loop by using a continue command to selectively stop this current iteration and move on to the next one.



12 Reductions

In many calculations, one of the computations you need to do is a reduction operation. This is where you take some list of values and reduce it down to a single value. In Python, you can use the command 'reduce(function, iterable)' to apply the reduction function to each pair of elements in the list. For example, if you apply the summation reduction operation to the list of the first five integers, you would get the result ((((1+2)+3)+4)+5). You can optionally add a third parameter to act as an initialisation term. It is loaded before any elements from the iterable, and is returned as a default if the iterable is actually empty. You can use a lambda function as the function parameter to reduce to keep your code as tight as possible. In this case, remember that it should only take two input parameters.

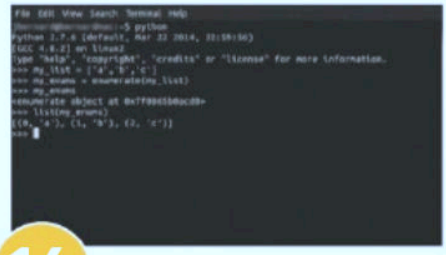
11 Filtering

Where the command map returns a result for every element in an iterable, filter only returns a result if the function returns a True value. This means that you can create a new list of elements where only the elements that satisfy some condition are used. As an example, if your function checked that the values were numbers between 0 and 10, then it would create a new list with no negative numbers and no numbers above 10. This could be accomplished with a for loop, but this method is much cleaner. If the function provided to filter is 'None', then it is assumed to be the identity function. This means that only those elements that evaluate to True are returned as part of the new list. There are iterable versions of filter available in the itertools module.

```
File  Edit  View  Search  Terminal  Help
                    :~$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> my_bools = [True, True, False, False]
>>> all(my_bools)
False
>>> any(my_bools)
True
>>> my_list = [0,1,2,3]
>>> all(my_list)
False
>>> any(my_list)
True
>>> my_list2 = ['a', 'b', 'c']
>>> all(my_list2)
True
>>> any(my_list2)
True
>>>
```

13 How true is a list?

In some cases, you may have collected a number of elements within a list that can be evaluated to True or False. For example, maybe you ran a number of possibilities through your computation and have created a list of which ones passed. You can use the command 'any(list)' to check to see whether any of the elements within your list are true. If you need to check whether all of the elements are True, you can use the command 'all(list)'. Both of these commands return a True if the relevant condition is satisfied, and a False if not. They do behave differently if the iterable object is empty, however. The command 'all' returns a True if the iterable is empty, whereas the command 'any' returns a False when given any empty iterable.



14 Enumerating

Sometimes, we need to label the elements that reside within an iterable object with their indices so that they can be processed at some later point. You could do this by explicitly looping through each of the elements and building an enumerated list. The enumerate command does this in one line. It takes an iterable object and creates a list of tuples as the result. Each tuple has the 0-based index of the element, along with the element itself. You can optionally start the indexing from some other value by including an optional second parameter. As an example, you could enumerate a list of names with the command 'list(enumerate(names, start=1))'. In this example, we decided to start the indexing at 1 instead of 0.

15 Casting

Variables in Python don't have any type information, and so can be used to store any type of object. The actual data, however, is of one type or another. Many operators, like addition, assume that the input values are of the same type. Very often, the operator you are using is smart enough to make the type of conversion that is needed. If you have the need to explicitly convert your data from one type to another, there are a class of functions that can be used to do this conversion process. The ones you are most likely to use is 'abs', 'bin', 'bool', 'chr', 'complex', 'float', 'hex', 'int', 'long', 'oct', and 'str'. For the number-based conversion functions, there is an order of precedence where some types are a subset of others. For example, integers are "lower" than floats. When converting up, no changes in the ultimate value should happen. When converting down, usually some amount of information is lost. For example, when converting from float to integer, Python truncates the number towards zero.

16 What is this?

Everything in Python is an object. You can check to see what class this object is an instance of with the command 'isinstance(object, class)'. This command returns a Boolean value.

17 Is it a subclass?

The command 'issubclass(class1, class2)' checks to see if class1 is a subclass of class2. If class1 and class2 are the same, this is returned as True.

18 Global objects

You can get a dictionary of the global symbol table for the current module with the command 'globals()'.

19 Local objects

You can access an updated dictionary of the current local symbol table by using the command 'locals()'.

20 Variables

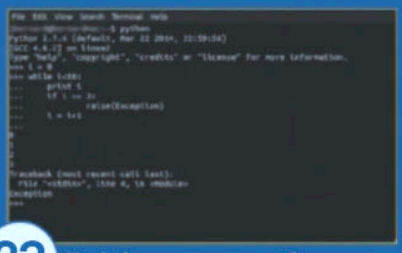
The command 'vars(dict)' returns writeable elements for an object. If you use 'vars()', it behaves like 'locals()'.

21 Making a global

A list of names can be interpreted as globals for the entire code block with the command 'global names'.

22 Nonlocals

In Python 3.X, you can access names from the nearest enclosing scope with the command 'nonlocal names' and bind it to the local scope.



23 Raising an exception

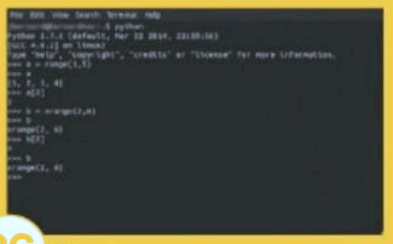
When you identify an error condition, you can use the 'raise' command to throw up an exception. You can include an exception type and a value.

24 Dealing with an exception

Exceptions can be caught in a try-except construction. If the code in the try block raises an exception, the code in the except block gets run.

25 Static methods

You can create a statis method, similar to that in Java or C++, with the command 'staticmethod(function_name)'.



26 Ranges

You may need a list of numbers, maybe in a 'for' loop. The command 'range()' can create an iterable list of integers. With one parameter, it goes from 0 to the given number. You can provide an optional start number, as well as a step size. Negative numbers count down.

27 Xranges

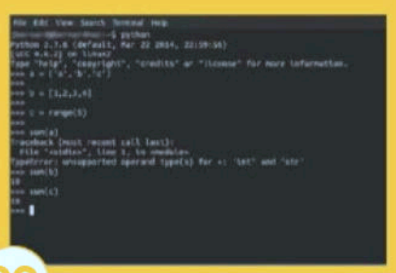
One problem with ranges is that all of the elements need to be calculated up front and stored in memory. The command 'xrange()' takes the same parameters and provides the same result, but only calculates the next element as it is needed.

28 Iterators

Iteration is a very Pythonic way of doing things. For objects which are not intrinsically iterable, you can use the command 'iter(object_name)' to essentially wrap your object and provide an iterable interface for use with other functions and operators.

29 Sorted lists

You can use the command 'sorted(list1)' to sort the elements of a list. You can give it a custom comparison function, and for more complex elements you can include a key function that pulls out a ranking property from each element for comparison.



30 Summing items

Above, we saw the general reduction function reduce. A specific type of reduction operation, summation, is common enough to warrant the inclusion of a special case, the command 'sum(iterable_object)'. You can include a second parameter here that will provide a starting value.

31 With modules

The 'with' command provides the ability to wrap a code block with methods defined by a context manager. This can help clean up code and make it easier to read what a given piece of code is supposed to be doing months later. A classic example of using 'with' is when dealing with files. You could use something like 'with open("myfile.txt", "r") as f:'. This will open the file and prepare it for reading. You can then read the file in the code block with 'data=f.read()'. The best part of doing this is that the file will automatically be closed when the code block is exited, regardless of the reason. So, even if the code block throws an exception, you don't need to worry about closing the file as part of your exception handler. If you have a more complicated 'with' example, you can create a context manager class to help out.

32 Printing

The most direct way of getting output to the user is with the print command. This will send text out to the console window. If you are using version 2.X of Python, there are a couple of ways you can use the print command. The most common way had been simply call it as 'print "Some text"'. You can also use print with the same syntax that you would use for any other function. So, the above example would look like 'print("Some text")'. This is the only form available in version 3.X. If you use the function syntax, you can add extra parameters that give you finer control over this output. For example, you can give the parameter 'file=myfile.txt' and get the output from the print command being dumped into the given text file. It also will accept any object that has some string representation available.

> "A classic example of using 'with' is when dealing with files. The best part of doing this is that the file will automatically be closed when the code block is exited, regardless of the reason"

```
File  Edit  View  Search  Terminal  Help
                    ~$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = "Hello World"
>>> b = memoryview(a)
>>> b
<memory at 0x7f7994f85938>
>>> list(b)
['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd']
>>> b[5]
' '
>>> b[6]
'W'
>>>
```

33 Memoryview

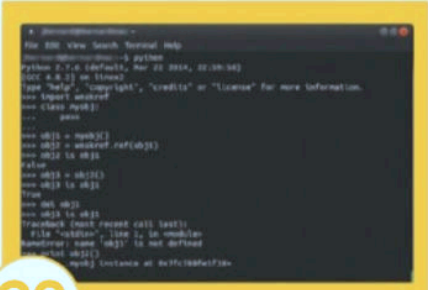
Sometimes, you need to access the raw data of some object, usually as a buffer of bytes. You can copy this data and put it into a bytearray, for example. But this means that you will be using extra memory, and this might not be an option for large objects. The command 'memoryview(object_name)' wraps the object handed in to the command and provides an interface to the raw bytes. It gives access to these bytes an element at a time. In many cases, elements are the size of one byte. But, depending on the object details, you could end up with elements that are larger than that. You can find out the size of an element in bytes with the property 'itemsize'. Once you have your memory view created, you can access the individual elements as you would get elements from a list (mem_view[1], for example).

34 Files

When dealing with files, you need to create a file object to interact with it. The file command takes a string with the file name and location and creates a file object instance. You can then call the file object methods like 'open', 'read' and 'close', to get data out of the file. If you are doing file processing, you can also use the 'readline' method. When opening a file, there is an explicit 'open()' command to simplify the process. It takes a string with the file name, and an optional parameter that is a string which defines the mode. The default is to open the file as read-only ('r'). You can also open it for writing ('w') and appending ('a'). After opening the file, a file object is returned so that you can further interact with it. You can then read it, write to it, and finally close it.

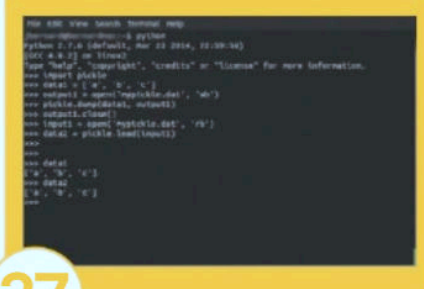
35 Yielding

In many cases, a function may need to yield the context of execution to some other function. This is the case with generators. The preferred method for a generator is that it will only calculate the next value when it is requested through the method 'next()'. The command 'yield' saves the current state of the generator function, and return execution control to the calling function. In this way, the saved state of the generator is reloaded and the generator picks up where it left off in order to calculate the next requested value. In this way, you only need to have enough memory available to store the bare minimum to calculate the next needed value, rather than having to store all of the possible values in memory all at once.



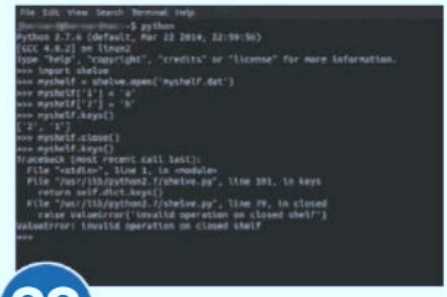
36 Weak references

You sometimes need to have a reference to an object, but still be able to destroy it if needed. A weak reference is one which can be ignored by the garbage collector. If the only references left to n object are weak references, then the garbage collector is allowed to destroy that object and reclaim the space for other uses. This is useful in cases where you have caches or mappings of large datasets that don't necessarily have to stay in memory. If an object that is weakly referenced ends up being destroyed and you try to access it, it will appear as a None. You can test for this condition and then reload the data if you decide that this is a necessary step.



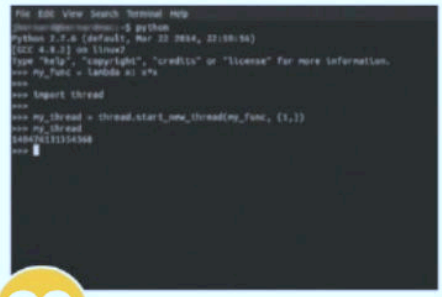
37 Pickling data

There are a few different ways of serialising memory when you need to checkpoint results to disk. One of these is called pickling. Pickle is actually a complete module, not just a single command. To store data on to the hard drive, you can use the dump method to write the data out. When you want to reload the same data at some other point in the future, you can use the load method to read the data in and unpickle it. One issue with pickle is its speed, or lack of it. There is a second module, cPickle, that provides the same basic functionality. But, since it is written in C, it can be as much as 1000 times faster. One thing to be aware of is that pickle does not store any class information for an object, but only its instance information. This means that when you unpickle the object, it may have different methods and attributes if the class definition has changed in the interim.



38 Shelving data

While pickling allows you save data and reload it, sometimes you need more structured object permanence in your Python session. With the shelve module, you can create an object store where essentially anything that can be pickled can be stored there. The backend of the storage on the drive can be handled by one of several systems, such as dbm or gdbm. Once you have opened a shelf, you can read and write to it using key value pairs. When you are done, you need to be sure to explicitly close the shelf so that it is synchronised with the file storage. Because of the way the data may be stored in the backing database, it is best to not open the relevant files outside of the shelve module in Python. You can also open the shelf with writeback set to True. If so, you can explicitly call the sync method to write out cached changes.



39 Threads

You can do multiple threads of execution within Python. The 'thread()' command can create a new thread of execution for you. It follows the same techniques as those for POSIX threads. When you first create a thread, you need to hand in a function name, along with whatever parameters said function needs. One thing to keep in mind is that these threads behave just like POSIX threads. This means that almost everything is the responsibility of the programmer. You need to handle mutex locks (with the methods 'acquire' and 'release'), as well as create the original mutexes with the method 'allocate_lock'. When you are done, you need to 'exit' the thread to ensure that it is properly cleaned up and no resources get left behind. You also have fine-grained control over the threads, being able to set things like the stack size for new threads.

40 Inputting data

Sometimes, you need to collect input from an end user. The command 'input()' can take a prompt string to display to the user, and then wait for the user to type a response. Once the user is done typing and hits the enter key, the text is returned to your program. If the readline module was loaded before calling input, then you will have enhanced line editing and history functionality. This command passes the text through eval first, and so may cause uncaught errors. If you have any doubts, you can use the command 'raw_input()' to skip this problem. This command simply returns the unchanged string inputted by the user. Again, you can use the readline module to get enhanced line editing.

```
File  Edit  View  Search  Terminal  Help
class my_class:
        _internal_num = 23
        _internal_string = "Hello Sir"
        def _internal_func():
                print "How did you find me?"
        def regular_func():
                print "You are allowed here"

my_obj = my_class()

my_obj._internal_num
~
~
~
~
~
~
```

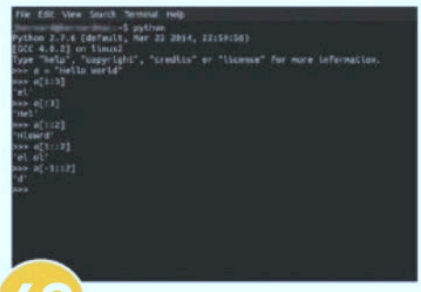
41 Internal variables

For people coming from other programming languages, there is a concept of having certain variables or methods be only available internally within an object. In Python, there is no such concept. All elements of an object are accessible. There is a style rule, however, that can mimic this type of behaviour. Any names that start with an underscore are expected to be treated as if they were internal names and to be kept as private to the object. They are not hidden, however, and there is no explicit protection for these variables or methods. It is up to the programmer to honour the intention from the author the class and not alter any of these internal names. You are free to make these types of changes if it becomes necessary, though.

```
File  Edit  View  Search  Terminal  Help
                    ~$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> str1 = "Hello World"
>>> str2 = str1
>>> str3 = "Hello World"
>>> str1 == str2
True
>>> str1 == str3
True
>>> cmp(str1, str2)
0
>>> cmp(str1, str3)
0
>>> str1 is str2
True
>>> str1 is str3
False
>>>
```

42 Comparing objects

There are several ways to compare objects within Python, with several caveats. The first is that you can test two things between objects: equality and identity. If you are testing identity, you are testing to see if two names actually refer to the same instance object. This can be done with the command 'cmp(obj1, obj2)'. You can also test this condition by using the 'is' keyword. For example, 'obj1 is obj2'. If you are testing for equality, you are testing to see whether the values in the objects referred to by the two names are equal. This test is handled by the operator '==', as in 'obj1 == obj2'. Testing for equality can become complex for more complicated objects.



43 Slices

While not truly a command, slices are too important a concept not to mention in this list of essential commands. Indexing elements in data structures, like lists, is one of the most common things done in Python. You can select a single element by giving a single index value. More interestingly, you can select a range of elements by giving a start index and an end index, separated by a colon. This gets returned as a new list that you can save in a new variable name. You can even change the step size, allowing you to skip some number of elements. So, you could grab every odd element from the list 'a' with the slice 'a[1::2]'. This starts at index 1, continues until the end, and steps through the index values 2 at a time. Slices can be given negative index values. If you do, then they start from the end of the list and count backwards.

"Python is an interpreted language, which means that the source code that you write needs to be compiled into a byte code format. This byte code then gets fed into the actual Python engine"

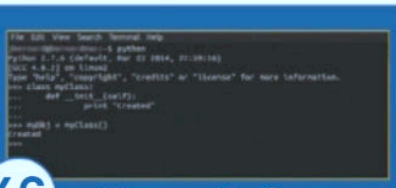
```
File  Edit  View  Search  Terminal  Help
                    :~$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> sqr1 = lambda x: x*x
>>>
>>> sqr1(10)
100
>>> sqr1(6)
36
>>> def gen_func(x):
...     return lambda y: y**x
...
>>> cubic = gen_func(3)
>>> cubic(2)
8
>>>
```

44 Lambda expressions

Since objects, and the names that point to them, are truly different things, you can have objects that have no references to them. One example of this is the lambda expression. With this, you can create an anonymous function. This allows you use functional programming techniques within Python. The format is the keyword 'lambda', followed by a parameter list, then a colon and the function code. For example, you could build your own function to square a number with 'lambda x: x*x'. You can then have a function that can programmatically create new functions and return them to the calling code. With this capability, you can create function generators to have self-modifying programs. The only limitation is that they are limited to a single expression, so you can't generate very complex functions.

45 Compiling code objects

Python is an interpreted language, which means that the source code that you write needs to be compiled into a byte code format. This byte code then gets fed into the actual Python engine to step through the instructions. Within your program, you may have the need to take control over the process of converting code to byte code and running the results. Maybe you wish to build your own REPL. The command 'compile()' takes a string object that contains a collection of Python code, and returns an object that represents a byte code translation of this code. This new object can then be handed in to either 'eval()' or 'exec()' to be actually run. You can use the parameter 'mode=' to tell compile what kind of code is being compiled. The 'single' mode is a single statement, 'eval' is a single expression and 'exec' is a whole code block.

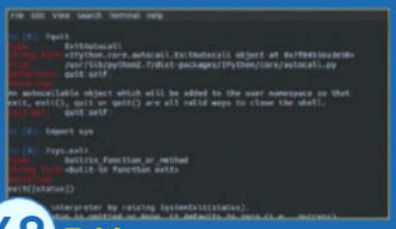


46 __init__ method

When you create a new class, you can include a private initialisation method that gets called when a new instance of the class is created. This method is useful when the new object instance needs some data loaded in the new object.

47 __del__ method

When an instance object is about to be destroyed, the __del__ method is called. This gives you the chance to do any kind of cleanup that may be required. This might be closing files, or disconnecting network connections. After this code is completed, the object is finally destroyed and resources are freed.

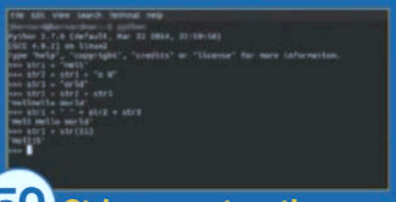


48 Exiting your program

There are two pseudo-commands available to exit from the Python interpreter: 'exit()' and quit()'. They both take an optional parameter which sets the exit code for the process. If you want to exit from a script, you are better off using the exit function from the sys module ('sys.exit(exit_code)'.

49 Return values

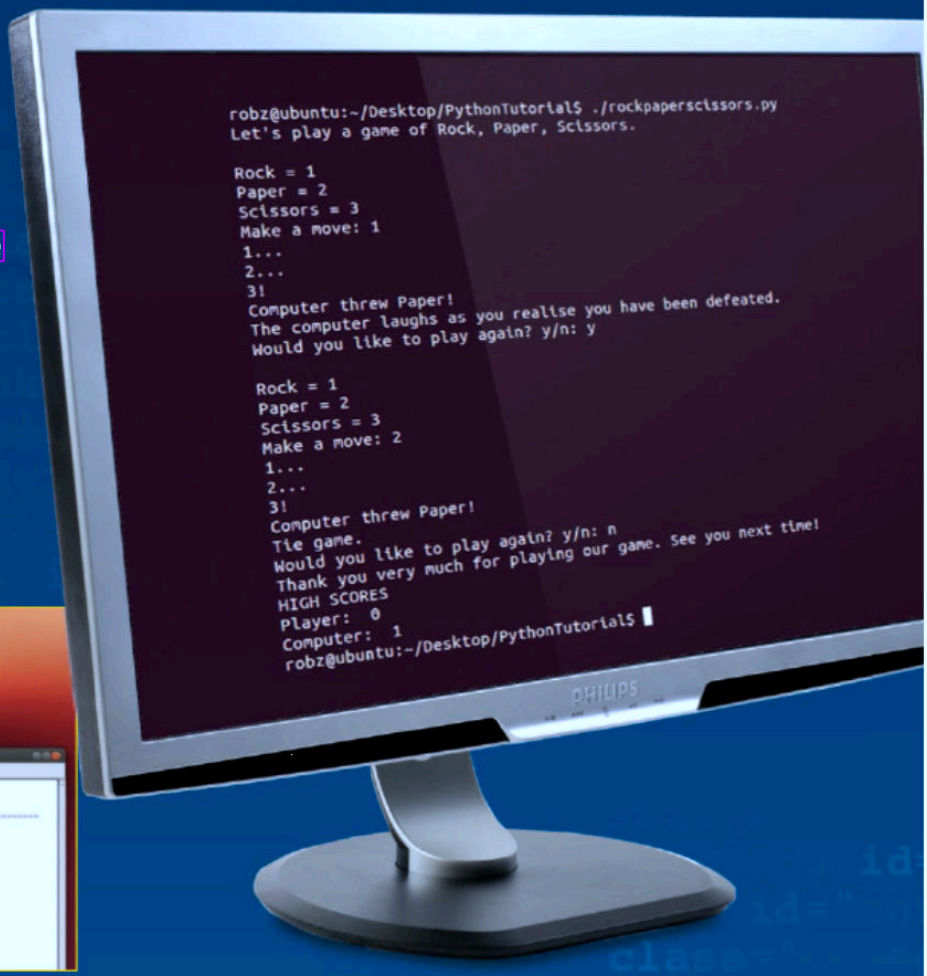
Functions may need to return some value to the calling function. Because essentially no name has a type, this includes functions. So functions can use the 'return' command to return any object to the caller.

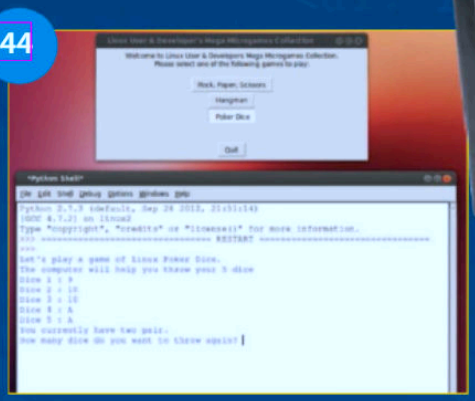


50 String concatenation

We will finish with what most lists start with – string concatenation. The easiest way to build up strings is to use the '+' operator. If you want to include other items, like numbers, you can use the 'str()' casting function to convert it to a string object.

Python Essentials





IDLE Colour Coding

Colour	Use for	Examples
Black	Data & variables	`23.6  area`
Green	Strings	`"Hello World"`
Purple	Functions	`len()  print()`
Orange	Commands	`if  for  else`
Blue	User functions	`get_area()`
Dark red	Comments	`#Remember VAT`
Light red	Error messages	`SyntaxError:`

Which is better Raspberry Pi or Arduino?

The **Raspberry Pi** is 40 times **faster** than an **Arduino** when it comes to clock speed. ... The **Raspberry Pi** is an independent computer that can run an actual operating system in Linux. It can multitask, support two USB ports, and connect wirelessly to the Internet. May 7, 2014

Arduino Vs. Raspberry Pi: Which Is The Right DIY Platform For You ...
https://readwrite.com/2014/05/07/arduino-vs-raspberry-pi-projects-diy-platform/

Search for: Which is better Raspberry Pi or Arduino?

Can I connect Arduino to Raspberry Pi?

In this tutorial, we **will connect** an **Arduino** to a **Raspberry Pi** and have the **Arduino** send “Hello from **Arduino**” to the **Raspberry Pi**, and the **Raspberry Pi will** blink an LED upon receiving the command from the **Arduino**. For communication, we **will** use simple serial communication over USB cable. Mar 23, 2018

How to Connect and Interface a Raspberry Pi With an Arduino ...
https://maker.pro/raspberry-pi/.../how-to-connect-and-interface-raspberry-pi-with-arduin...

Search for: Can I connect Arduino to Raspberry Pi?

Can Raspberry Pi be used as a microcontroller?

While the **Raspberry Pi** is a versatile computer, the Arduino board is a highly focused **microcontroller**. The Arduino has 14 digital input/output pins (female header) of which six **can** be **used** as PWM (pulse-width modulation) outputs to control devices such as servos, RGB light levels, and other devices that need precision.

Arduino and Raspberry Pi: using a microcontroller in your projects ...
https://www.raspberrypi.org/magpi/arduino-raspberry-pi/

Search for: Can Raspberry Pi be used as a microcontroller?

Which Arduino is best?

Let us first see the list of Arduino boards and will get to the point later:

- Arduino Diecimila, Duemilanove, UNO, Nano, Mini - Atmel 328p/ 8-bit.
- Mega and Mega 2560 R3 version - Atmel 2560/ 8-bit.
- Leonardo, Micro, LilyPad- Atmel 32U4/ 8-bit.
- Arduino Due - Arm Cortex M3/ 32-bit.
- Arduino Zero - Arm Cortex M0+ core/ 32-bit.

Which is the best arduino board for a beginner? Between the UNO ...
https://www.quora.com/Which-is-the-best-arduino-board-for-a-beginner-Between-the-U...

Search for: Which Arduino is best?

Can we use Raspberry Pi instead of Arduino?

What programming language does Raspberry Pi use?

Is Raspberry Pi real time?

ChibiOS/RT on the **Raspberry Pi**. ChibiOS/RT is a portable **real-time** operating system (RTOS) designed for embedded applications. Although the **Raspberry Pi** is typically used with the Linux operating system, it's not necessarily the best choice for applications requiring low latency, predictable response to external events ...

Getting Started With ChibiOS/RT on the Raspberry Pi - Steve Bate
https://www.stevebate.net/chibios-rpi/GettingStarted.html

Search for: Is Raspberry Pi real time?

Why is it called Raspberry Pi?

Raspberry is a reference to a fruit naming tradition in the old days of microcomputers. ... There's Tangerine Computer Systems, Apricot Computers, and the old British company Acorn, which is a family of fruit. **Pi** is because originally we were going to produce a computer that could only really run Python. May 22, 2012

Interview with Raspberry's Founder Eben Upton - TechSpot
https://www.techspot.com/article/531-eben-upton-interview/

Search for: Why is it called Raspberry Pi?

Is Arduino a microcontroller?

"**Arduino**" is a software development environment and any of several **microcontroller** boards that the software environment can develop programs for. Most of the boards use Atmel AVR **microcontrollers**. **Arduino** is a **microcontroller** based platform (ATMEGA 328 for the UNO). Feb 12, 2014

Arduino vs Microprocessor vs Microcontroller - Electrical ...
https://electronics.stackexchange.com/.../arduino-vs-microprocessor-vs-microcontroller

Search for: Is Arduino a microcontroller?

Is Raspberry Pi compatible with Arduino?

Can Raspberry Pi run Arduino code?

Can Raspberry Pi run Arduino code?

Raspberry Pi is an amazing minicomputer, and I would love to use it in some projects. ... That **will** allow us to compile the **Arduino code** into binaries which **can run** on **Raspberry Pi**. But before we **can** do that, we have to prepare a few things, both in the **Arduino** IDE and on **Raspberry Pi**. Feb 21, 2018

How to Run Arduino Sketches on Raspberry Pi - Device Plus
https://www.deviceplus.com/how-tos/.../how-to-run-arduino-sketches-on-raspberry-pi/

Search for: Can Raspberry Pi run Arduino code?

How do I program my Raspberry Pi with Arduino?

This will take very little time and will get your Raspberry Pi to program your Arduino.

1. Step 1: Boot up your Raspberry Pi. Connect your Raspberry Pi to a power source and a monitor. ...
2. Step 2: Install the IDE. Once you open up the terminal, type these commands:
3. Step 3: Reboot. ...
4. Step 4: Using the Arduino IDE.

Oct 13, 2017

Program Your Arduino From Your Raspberry Pi - Hackster.io
https://www.hackster.io/techno.../program-your-arduino-from-your-raspberry-pi-3407d...

Search for: How do I program my Raspberry Pi with Arduino?

What language does Arduino use?

In fact, you already are; the Arduino language is merely a set of **C/C++** functions that can be called from your code. Your sketch undergoes minor changes (e.g. automatic generation of function prototypes) and then is passed directly to a **C/C++ compiler** (**avr**-g++).

Arduino - FAQ
https://www.arduino.cc/en/main/FAQ

Search for: What language does Arduino use?

Is Arduino good for beginners?

There are many sketches that will run on the Uno, allowing the beginner to evaluate Arduino and use it as a learning platform. Many small projects can be built using an Arduino Uno. If you have specific need for an Arduino with more memory, and/or pins, then the **MEGA 2560** is a good choice. May 11, 2017

Which Arduino for Beginners | Choosing an Arduino

Yes. **Python can** be **used** to program an **Arduino**, simply by importing pyfirmata, which **can** interface the **arduino** with **Python**. **I**'m affraid **Arduino** IDE doesn't **use** C/C++ language. It's derived from Processing IDE and has quite simplified C/Java-like syntax. Feb 13, 2014

Programming an Arduino using Python, rather than C/C++ - Arduino ...
https://arduino.stackexchange.com/.../programming-an-arduino-using-python-rather-tha...

Search for: Can you use Python with Arduino?

How does the Arduino work?

Does Raspberry Pi need fan?

Is Raspberry Pi an embedded system?

Is Raspberry Pi microcontroller?

Is Arduino real time?

Whats is Raspberry Pi?

What is the point of Raspberry Pi?

Is the Raspberry PI 3 worth it?

Do Arduino sensors work with Raspberry Pi?

What is IoT using Raspberry Pi?

What can I do with a Raspberry Pi?

Can you power a Raspberry Pi from GPIO?

Is Elegoo the same as Arduino?

The **Elegoo** uno works the **same** as the **Arduino** brand. its just not the official "Name Brand". some of the hardware on the board will look different, but do not be worried. Matter of fact, the kit actually has you download **Arduinos** software and drivers to Run/Program the **Elegoo** uno .

Amazon.com: Customer Questions & Answers
https://www.amazon.com/ask/questions/Tx2363FRDI9U5UN

Search for: Is Elegoo the same as Arduino?

What are the types of Arduino?

What are the Different Types Of Arduino Boards

- Types Of Arduino Boards.
- Arduino Uno (R3)
- LilyPad Arduino Board.
- RedBoard Arduino Board.
- Arduino Mega (R3) Board.
- Arduino Leonardo Board.

ARDUINO CHEAT SHEET

JEROEN DOGGEN, AP UNIVERSITY COLLEGE ANTWERP

ARTESIS PLANTIJN HOGESCHOOL ANTWERPEN

Structure

void **setup**()
void **loop**()

Control Structures

```
if(x<5){}
for(int i = 0; i < 255; i++ ){}
while((x < 6 ){}
```

Further Syntax

// Single line comment
/* .. */ Multi line comment
#define ANSWER 42
#include <myLib.h>

General Operators

=	assignment
+, −	addition, substraction
*, /	multiplication, division
%	modulo
==	equal to
!=	not equal to
<	less than
<=	less than or equal to

Pointer Access

&	reference operator
*	dereference operator

Bitwise Operators

&	bitwise AND
|	bitwise OR
^	bitwise XOR
~	bitwise NOT

Compound Operators

++	Increment
−−	Decrement
+=	Compound addition
&=	Compound bitwise AND

Constants

```
HIGH, LOW
INPUT, OUTPUT
true, false
```

53 : Decimal
B11010101 : Binary
0x5BA4 : Hexadecimal

Data Types

void	
boolean	0, 1, false, true
char	e.g. 'a' -128 → 127
unsigned char	0 → 255
int	-32.768 → 32.767
unsigned int	0 → 65535
long	-2.147.483.648 → 2.147.483.647
float	-3,4028235E+38 → 3.402835E+38
sizeof (myint)	returns 2 bytes

Arrays

int myInts[6];
int myPins[]=2,4,8,5,6;
int myVals[6]=2,-4,9,3,5;

Strings

char S1[15];
char S2[8]='A','r','d','u','i','n','o';
char S3[8]='A','r','d','u','i','n','o','\0';
char S4[]="Arduino";
char S5[8] = "Arduino";
char S6[15] = "Arduino";

Conversion

char()	**int()**	**long()**
byte()	**word()**	**float()**

Qualifiers

static	Persist between calls
volatile	Use RAM (nice for ISR)
const	Mark read-only
PROGMEM	Use flash memory

Interrupts

attachInterrupt(interrupt, function, type)
detachInterrupt(interrupt)
boolean(interrupt)
interrupts()
noInterrupts()

Advanced I/O

tone(pin, freqhz)
tone(pin, freqhz, duration_ms)
noTone(pin)
shiftOut (dataPin, clockPin, how, value)
unsigned long **pulseIn**(pin, [HIGH,LOW])

Time

unsigned long **millis()**	50 days overflow
unsigned long **micros()**	70 min overflow

delay(ms)
delayMicroseconds(us)

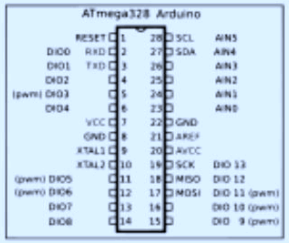
Math

min(x,y) **max**(x,y) **abs**(x)
sin(rad) **cos**(rad) **tan**(rad)
pow(base, exponent)
map(val, fromL, fromH, toL, toH)
constrain(val, fromL, toH)

Pseudo Random Numbers

randomSeed(seed)
long **random**(max)
long **random**(min, max)

ATmega328 Pinout



I/O Pins

	Uno	Mega
# of IO	14 + 6	54 + 11
Serial Pins 3	0 - RX, 1 -TX	RX1 → RX4
Interrupts	2,3	2,3,18,19,20,21
PWM Pins	5,6 - 9,10 - 3,11	0 → 13
SPI (SS, MOSI, MISO, SCK)	10→ 13	50→ 53
I2C (SDA, SCK)	A4, A5	20,21

Analog I/O

```
analogReference(EXTERNAL, INTERNAL)
analogRead(pin)
analogWrite(pin, value)
```

Digital I/O

```
pinMode(pin,[INPUT,OUTPUT])
digitalRead(pin)
digitalWrite(pin, value)
```

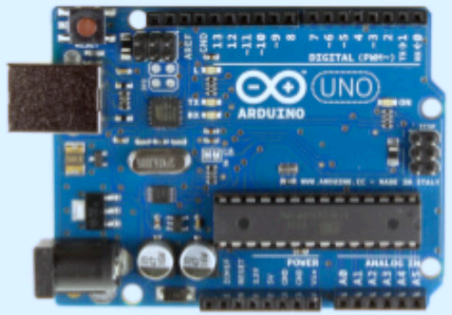
Serial Communication

Serial.begin(speed)
Serial.print("Text")
Serial.println("Text")

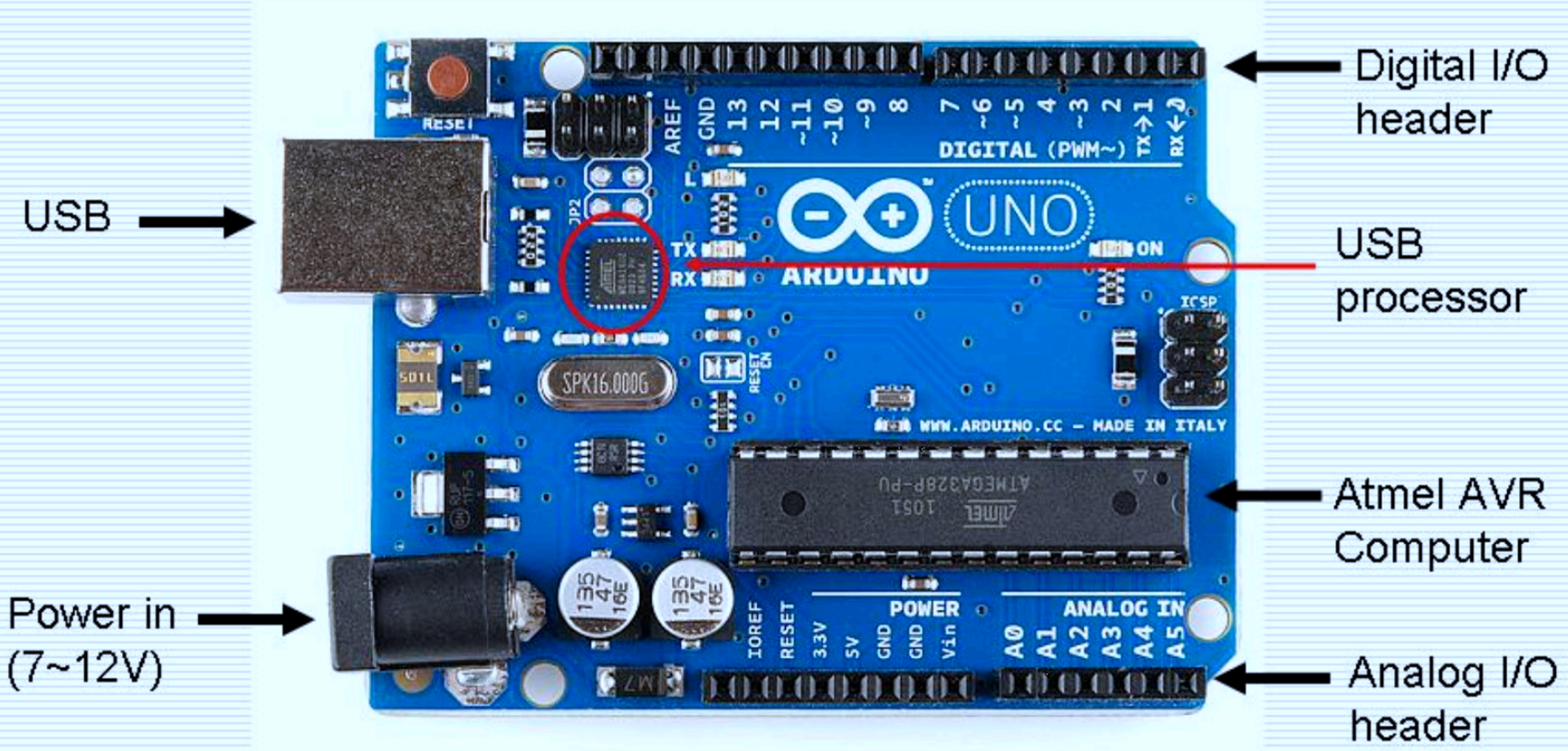
Websites

forum.arduino.cc
playground.arduino.cc
arduino.cc/en/Reference

Arduino Uno Board



What's an Arduino?



Microprocessor board with all the support electronics built-in on the board. Has 14 digital I/O and 6 analog I/O connections. USB port, power jack, and voltages for support electronics attached to the board via the headers.

Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference
http://arduino.cc/en/Reference/

Structure & Flow

Basic Program Structure

```
void setup() {
  // Runs once when sketch starts
}
void loop() {
  // Runs repeatedly
}
```

Control Structures

```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break;     // Exit a loop immediately
continue;  // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x;  // x must match return type
return;    // For void return type
```

Function Definitions

```
<ret. type> <name>(<params>) { ... }
e.g. int double(int x) {return x*2;}
```

Operators

General Operators

=	assignment		
+	add	-	subtract
*	multiply	/	divide
%	modulo		
==	equal to	!=	not equal to
<	less than	>	greater than
<=	less than or equal to		
>=	greater than or equal to		
&&	and	||	or
!	not		

Compound Operators

++	increment
--	decrement
+=	compound addition
-=	compound subtraction
*=	compound multiplication
/=	compound division
&=	compound bitwise and
|=	compound bitwise or

Bitwise Operators

&	bitwise and	|	bitwise or
^	bitwise xor	~	bitwise not
<<	shift left	>>	shift right

Pointer Access

&	reference: get a pointer
*	dereference: follow a pointer

Variables, Arrays, and Data

Data Types

boolean	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	i.e., no return value

Strings

```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

Arrays

```
int myPins[] = {2, 4, 8, 3, 6};
int myInts[6];   // Array of 6 ints
myInts[0] = 42;  // Assigning first
                 // index of myInts
myInts[6] = 12;  // ERROR! Indexes
                 // are 0 though 5
```

Built-in Functions

Pin Input/Output

Digital I/O - pins 0-13 A0-A5

```
pinMode(pin,
  [INPUT, OUTPUT, INPUT_PULLUP])
int digitalread(pin)
digitalWrite(pin, [HIGH, LOW])
```

Analog In - pins A0-A5

```
int analogRead(pin)
analogReference(
  [DEFAULT, INTERNAL, EXTERNAL])
```

PWM Out - pins 3 5 6 9 10 11

```
analogWrite(pin, value)
```

Advanced I/O

```
tone(pin, freq_Hz)
tone(pin, freq_Hz, duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
  [MSBFIRST, LSBFIRST], value)
unsigned long pulseIn(pin,
  [HIGH, LOW])
```

Time

```
unsigned long millis()
  // Overflows at 50 days
unsigned long micros()
  // Overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

Math

```
min(x, y)   max(x, y)   abs(x)
sin(rad)    cos(rad)    tan(rad)
sqrt(x)     pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers

```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes

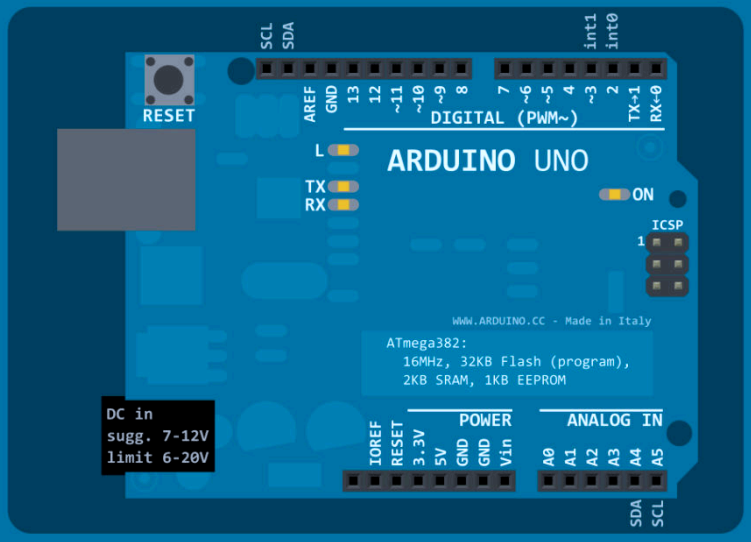
```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Type Conversions

```
char(val)     byte(val)
int(val)      word(val)
long(val)     float(val)
```

External Interrupts

```
attachInterrupt(interrupt, func,
 [LOW, CHANGE, RISING, FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```



Libraries

Serial - comm. with PC or via RX/TX

```
begin(long speed)  // Up to 115200
end()
int available()  // #bytes available
int read()   // -1 if none available
int peek()   // Read w/o removing
flush()
print(data)    println(data)
write(byte)    write(char * string)
write(byte * data, size)
SerialEvent()  // Called if data rdy
```

SoftwareSerial.h - comm. on any pin

```
SoftwareSerial(rxPin, txPin)
begin(long speed)  // Up to 115200
listen()       // Only 1 can listen
isListening()  // at a time.
read, peek, print, println, write
  // Equivalent to Serial library
```

EEPROM.h - access non-volatile memory

```
byte read(addr)
write(addr, byte)
EEPROM[index]  // Access as array
```

Servo.h - control servo motors

```
attach(pin, [min_uS, max_uS])
write(angle)  // 0 to 180
writeMicroseconds(uS)
   // 1000-2000; 1500 is midpoint
int read()    // 0 to 180
bool attached()
detach()
```

Wire.h - I²C communication

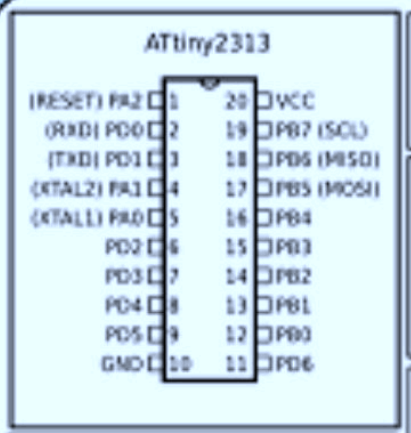
```
begin()     // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte)              // Step 2
send(char * string)
send(byte * data, size)
endTransmission()       // Step 3
int available() // #bytes available
byte receive()  // Get next byte
onReceive(handler)
onRequest(handler)
```

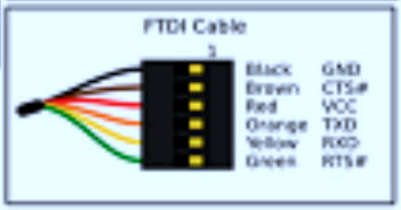

by Mark Liffiton

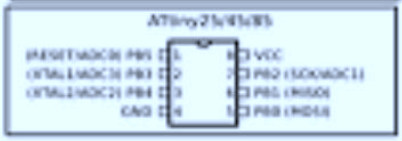
Adapted from:
- Original: Gavin Smith
- SVG version: Frederic Dufourg
- Arduino board drawing: Fritzing.org

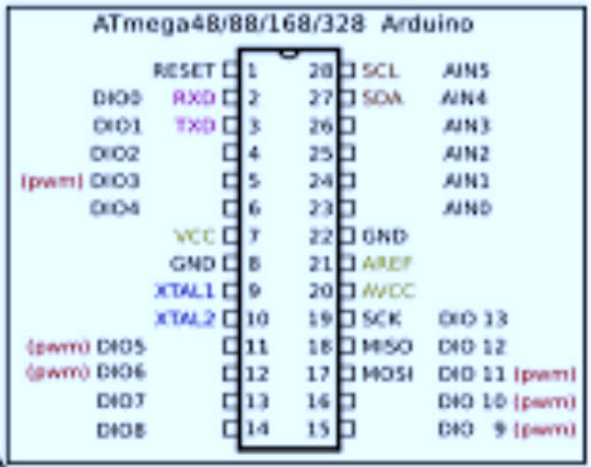
	ATMega168	ATMega328	ATmega1280
Flash (2k for bootloader)	16kB	32kB	128kB
SRAM	1kB	2kB	8kB
EEPROM	512B	1kB	4kB

	Duemilanove/ Nano/ Pro/ ProMini	Mega
# of IO	14 + 6 analog (Nano has 14+8)	54 + 16 analog
Serial Pins	0 - RX 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 3 - (Int 1)	2,3,21,20,19,18 (IRQ0- IRQ5)
PWM pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0-13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL









From Arduino.CC

Arduino Cheat Sheet V.02b

Mostly taken from the extended reference:
http://arduino.cc/en/Reference/Extended
Gavin Smith – Robots and Dinosaurs, The Sydney Hackspace

Structure

```
void setup()   void loop()
```

Control Structures

```
if (x<5){  } else {  }
switch (myvar) {
  case 1:
    break;
  case 2:
    break;
  default:
}
for (int i=0; i <= 255; i++){   }
while (x<5){  }
do {  } while (x<5);
continue; //Go to next in do/for/while loop
return x; // Or 'return;' for voids.
goto          // considered harmful :-)
```

Further Syntax

```
// (single line comment)
/*  (multi-line comment) */
#define DOZEN 12 //Not baker's!
#include <avr/pgmspace.h>
```

General Operators

```
= (assignment operator)
+  (addition)          - (subtraction)
* (multiplication)  / (division)
% (modulo)
== (equal to)        != (not equal to)
< (less than)        > (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and)      || (or)      ! (not)
```

Pointer Access

```
& reference operator
* dereference operator
```

Bitwise Operators

```
& (bitwise and)  | (bitwise or)
^ (bitwise xor) ~ (bitwise not)
<< (bitshift left)   >> (bitshift right)
```

Compound Operators

```
++ (increment)     -- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)
```

Constants

```
HIGH | LOW
INPUT | OUTPUT
true | false
143 // Decimal number
0173 // Octal number
B11011111 //Binary (8-bits only)
0x7B // Hex number
7U // Force unsigned
10L // Force long
15UL // Force long unsigned
10.0 // Forces floating point
2.4e5 // 245,000
```

Data Types

```
void
boolean          (0, 1, false, true)
char   (e.g. 'a' -128 to 127)
unsigned char (0 to 255)
byte   (0 to 255)
int      (-32,768 to 32,767)
unsigned int (0 to 65535)
word   (0 to 65535)
long          (-2,147,483,648 to
               2,147,483,647)
unsigned long (0 to 4,294,967,295)
float         (-3.4028235E+38 to
               3.4028235E+38)
double (currently same as float)
sizeof(myint) // returns 2 bytes
```

Strings

```
char S1[15];
char S2[8]={'a','r','d','u','i','n','o'};
char S3[8]={'a','r','d','u','i','n','o','\0'};
  //Included \0 null termination
char S4[ ] = "arduino";
char S5[8] = "arduino";
char S6[15] = "arduino";
```

Arrays

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
```

Conversion

```
char()          byte()
int()           word()
long()          float()
```

Qualifiers

```
static  // persists between calls
volatile    // use  RAM (nice for ISR)
const  // make read-only
PROGMEM // use flash
```

Digital I/O

```
pinMode(pin, [INPUT,OUTPUT])
digitalWrite(pin, value)
int digitalRead(pin)
//Write High to inputs to use pull-up res
```

Analog I/O

```
analogReference([DEFAULT,INTERNAL,EXTERNAL])
int analogRead(pin) //Call twice if
switching pins from high Z source.
analogWrite(pin, value) // PWM
```

Advanced I/O

```
tone(pin, freqhz)
tone(pin, freqhz ,duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
[MSBFIRST,LSBFIRST], value)
unsigned long pulseIn(pin, [HIGH,LOW])
```

Time

```
unsigned long millis() // 50 days overflow.
unsigned long micros() // 70 min overflow
delay(ms)
delayMicroseconds(us)
```

Math

```
min(x, y)     max(x, y)      abs(x)
constrain(x, minval, maxval )
map(val, fromL, fromH, toL, toH)
pow(base, exponent)  sqrt(x)
sin(rad)      cos(rad)       tan(rad)
```

Random Numbers

```
randomSeed(seed) // Long or int
long random(max)
long random(min, max)
```

Bits and Bytes

```
lowByte()          highByte()
bitRead(x,bitn)    bitWrite(x,bitn,bit)
bitSet(x,bitn)     bitClear(x,bitn)
bit(bitn) //bitn:  0-LSB 7-MSB
```

External Interrupts

```
attachInterrupt(interrupt, function,
[LOW,CHANGE,RISING,FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

Libraries:

Serial.

```
  begin([300, 1200, 2400, 4800, 9600,
14400, 19200, 28800, 38400, 57600,
115200])
  end()
  int available()
  int read()
  flush()
  print()
  println()
  write()
```

EEPROM (#include <EEPROM.h>)

```
  byte read(intAddr)
       write(intAddr,myByte)
```

Servo (#include <Servo.h>)

```
  attach(pin , [min_uS, max_uS])
  write(angle) // 0-180
  writeMicroseconds(uS) //1000-2000,
   1500 is midpoint
  read() // 0-180
  attached() //Returns boolean
  detach()
```

SoftwareSerial(RxPin,TxPin)

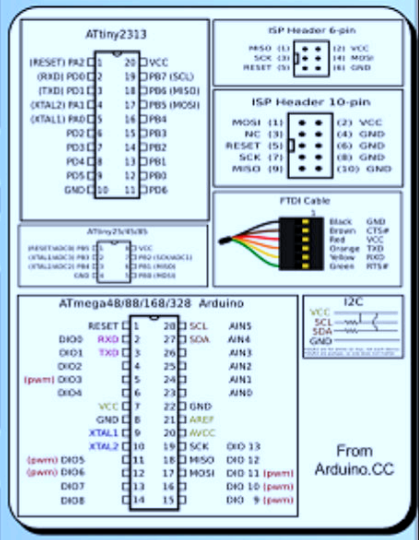
```
  // #include<SoftwareSerial.h>
  begin(longSpeed) // up to 9600
  char read() // blocks till data
  print(myData) or println(myData)
```

Wire (#include <Wire.h>) // For **I2C**

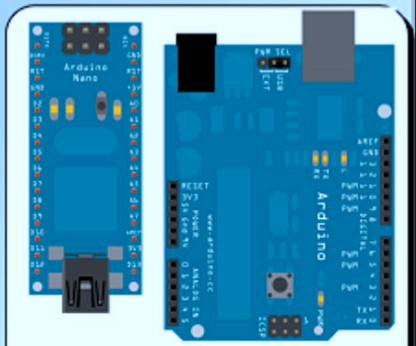
```
  begin()          // Join as master
  begin(addr) // Join as slave @ addr
  requestFrom(address, count)
  beginTransmission(addr) // Step 1
  send(mybyte)          // Step 2
  send(char * mystring)
  send(byte * data, size)
  endTransmission()       // Step 3
  byte available() // Num of bytes
  byte receive() //Return next byte
  onReceive(handler)
  onRequest(handler)
```

	ATMega168	ATMega328	ATmega1280
Flash (2k for bootloader)	16kB	32kB	128kB
SRAM	1kB	2kB	8kB
EEPROM	512B	1kB	4kB

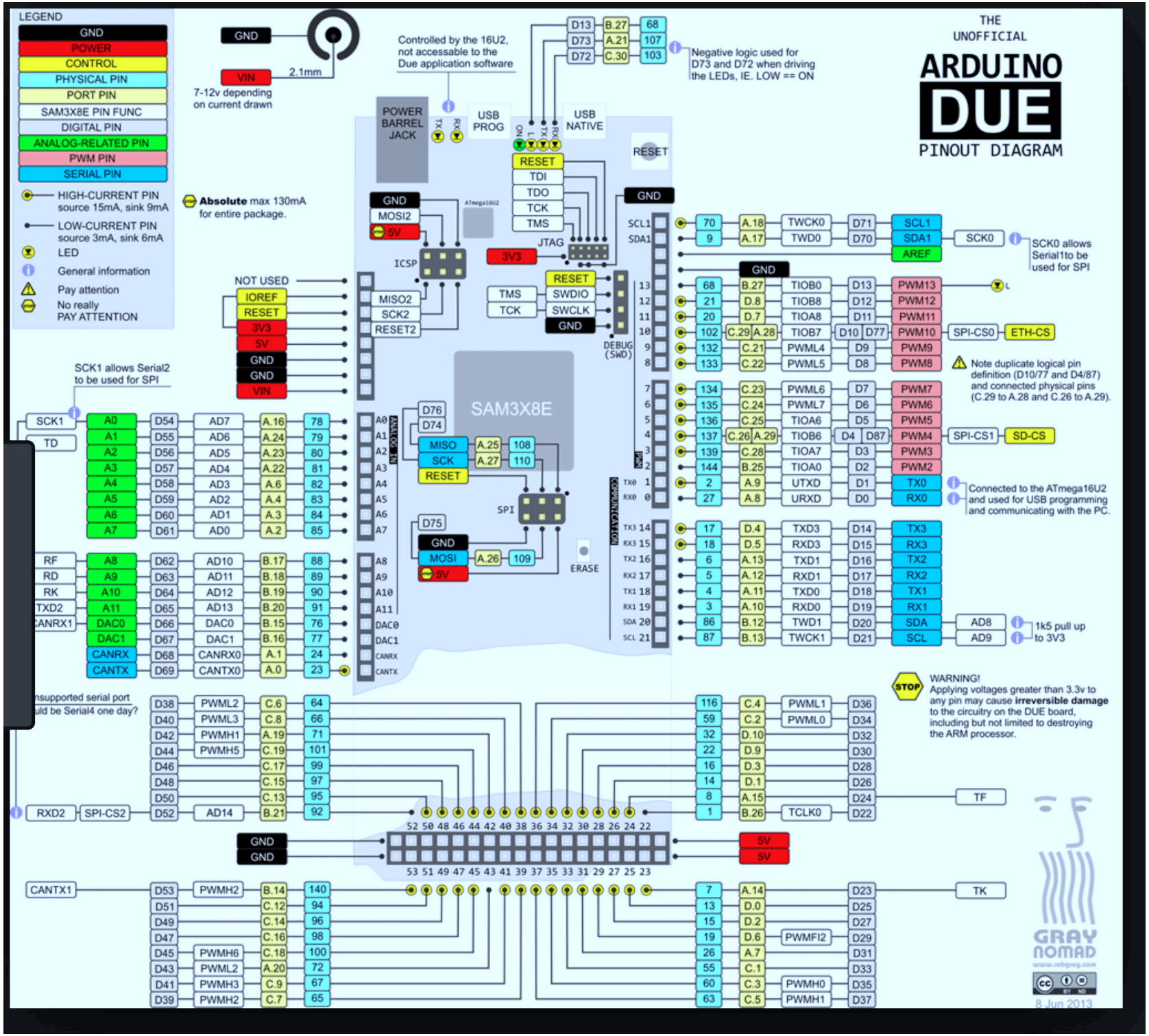
	Duemilanove/ Nano/ Pro/ ProMini	Mega
# of IO	14 + 6 analog (Nano has 14+8)	54 + 16 analog
Serial Pins	0 - RX 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 3 - (Int 1)	2,3,21,20,19,18 (IRQ0- IRQ5)
PWM pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0-13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL



From Arduino.CC



Pics from Fritzing.Org under C.C. license


THE UNOFFICIAL
ARDUINO DUE
PINOUT DIAGRAM
LEGEND
GND
POWER
CONTROL
PHYSICAL PIN
PORT PIN
SAM3X8E PIN FUNC
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
HIGH-CURRENT PIN source 15mA, sink 9mA
LOW-CURRENT PIN source 3mA, sink 6mA
LED
General information
Pay attention
No really PAY ATTENTION
GND
VIN
2.1mm
7-12v depending on current drawn
Absolute max 130mA for entire package.
Controlled by the 16U2, not accessable to the Due application software
Negative logic used for D73 and D72 when driving the LEDs, IE. LOW == ON
POWER BARREL JACK
USB PROG
USB NATIVE
RESET
TDI
TDO
TCK
TMS
JTAG
3V3
ICSP
MISO2
SCK2
RESET2
SWDIO
SWCLK
DEBUG (SWD)
NOT USED
IOREF
5V
SAM3X8E
SCK1 allows Serial2 to be used for SPI
SCK1
TD
SPI
ERASE
SCK0 allows Serial1to be used for SPI
SCK0
SCL1
SDA1
AREF
SPI-CS0
ETH-CS
SPI-CS1
SD-CS
Note duplicate logical pin definition (D10/77 and D4/87) and connected physical pins (C.29 to A.28 and C.26 to A.29).
Connected to the ATmega16U2 and used for USB programming and communicating with the PC.
1k5 pull up to 3V3
WARNING!
Applying voltages greater than 3.3v to any pin may cause irreversible damage to the circuitry on the DUE board, including but not limited to destroying the ARM processor.
TF
TK
GRAY NOMAD
8 Jun 2013

<!DOCTYPE>	Version of (X)HTML
<html>	HTML document
<head>	Page information
<body>	Page contents

Comments

<!-- Comment Text -->

Page Information

<base />	Base URL
<meta />	Meta data
<title>	Title
<link />	Relevant resource
<style>	Style resource
<script>	Script resource

Document Structure

<h[1-6]>	Heading
	Page section
	Inline section
<p>	Paragraph
 	Line break
<hr />	Horizontal rule

Links

<a href="">	Page link
<a href="mailto:">	Email link
<a name="name">	Anchor
<a href="#name">	Link to anchor

Text Markup

	Strong emphasis
	Emphasis
<blockquote>	Long quotation
<q>	Short quotation
<abbr>	Abbreviation
<acronym>	Acronym
<address>	Address
<pre>	Pre-formatted text
<dfn>	Definition
<code>	Code
<cite>	Citation
<del>	Deleted text
<ins>	Inserted text
<sub>	Subscript
<sup>	Superscript
<bdo>	Text direction

	Ordered list
	Unordered list
	List item
<dl>	Definition list
<dt>	Definition term
<dd>	Term description

Forms

<form>	Form
<fieldset>	Collection of fields
<legend>	Form legend
<label>	Input label
<input />	Form input
<select>	Drop-down box
<optgroup>	Group of options
<option>	Drop-down options
<textarea>	Large text input
<button>	Button

Tables

<table>	Table
<caption>	Caption
<thead>	Table header
<tbody>	Table body
<tfoot>	Table footer
<colgroup>	Column group
<col />	Column
<tr>	Table row
<th>	Header cell
<td>	Table cell

Images and Image Maps

<img />	Image
<map>	Image Map
<area />	Area of Image Map

Common Character Entities

"	"	Quotation mark
&	&	Ampersand
<	<	Less than
>	>	Greater than
@	@	"At" symbol
€	€	Euro
•	•	Small bullet
™	™	Trademark
£	£	Pound
		Non-breaking space
©	©	Copyright symbol

<object>	Object
<param />	Parameter

Empty Elements

<area />	<img />
<base />	<input />
 	<link />
<col />	<meta />
<hr />	<param />

Core Attributes

class	style
id	title

Note: Core Attributes may not be used in base, head, html, meta, param, script, style or title elements.

Language Attributes

dir	lang

Note: Language Attributes may not be used in base, br, frame, frameset, hr, iframe, param or script elements.

Keyboard Attributes

accesskey	tabindex

Window Events

onLoad	onUnload

Form Events

onBlur	onReset
onChange	onSelect
onFocus	onSubmit

Keyboard Events

onKeydown	onKeyup
onKeypress	

Mouse Events

onClick	onMouseout
onDblclick	onMouseover
onMousedown	onMouseup
onMousemove	

HTML QUICK SHEET

Arduino

http://arduino.cc

Blink example

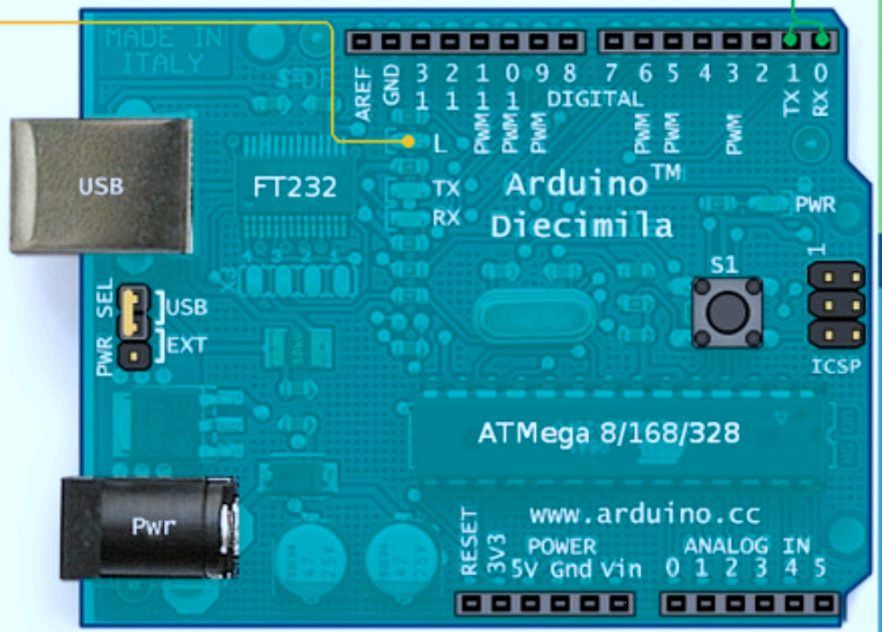
```
/*
 * Blink
 *
 * http://arduino.cc/en/Tutorial/Blink
 */

int ledPin =  13;   // LED on digital pin 13

// setup()  runs once, when sketch starts
void setup()   {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

 loop()  runs over and over again
id loop()

 igitalWrite(ledPin, HIGH); // set the LED on
 elay(1000);                // wait for a second
 igitalWrite(ledPin, LOW);  // set the LED off
 elay(1000);                // wait for a second
```

Constants

```
efine HIGH      0x1     // 3-5V
efine LOW       0x0     // 0-2V

#define INPUT   0x0
#define OUTPUT  0x1
```

Time

```
unsigned long millis();
unsigned long micros();
delay(ms);
delayMicroseconds(us);
```



Atmega168 Pin Mapping

Arduino function					Arduino function
reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22	GND	GND
GND	GND	8	21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)

Interrupts

```
attachInterrupt(interrupt, function, mode);
detachInterrupt(interrupt);
                                    interrupts();
                                    noInterrupts();
```

Serial

```
                Serial.begin( 9600 );
count =         Serial.available();
char  =         Serial.read();
                Serial.write(value);
                Serial.flush();
                Serial.print(b, BYTE); //DEC,HEX,OCT,BIN
                Serial.println(value);
                Serial.end();
```

Functions I/O

```
Digital
pinMode(pin, mode); // OUTPUT, INPUT
digitalWrite(pin, value);
int digitalRead(pin);
                    Analog
                    int analogRead(pin);
                    analogWrite(pin, value); // PWM


shiftOut(dataPin, clockPin, bitOrder, value);
unsigned long pulseIn(pin, value);
```

Math

```
min(x, y);
max(x, y);
abs(x);
constrain(x, a, b);
map(value, fromLow, fromHigh, toLow, toHigh);
pow(base, exponent);
sq(x);
sqrt(x);
sin(rad);
cos(rad);
tan(rad);
```

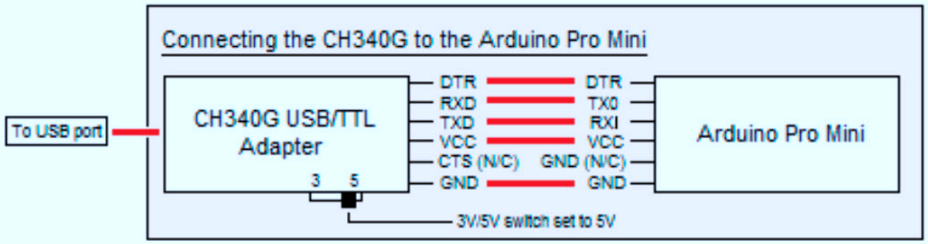
```
                    Random
                    randomSeed(seed);
                    long random(max);
                    long random(min, max);
```

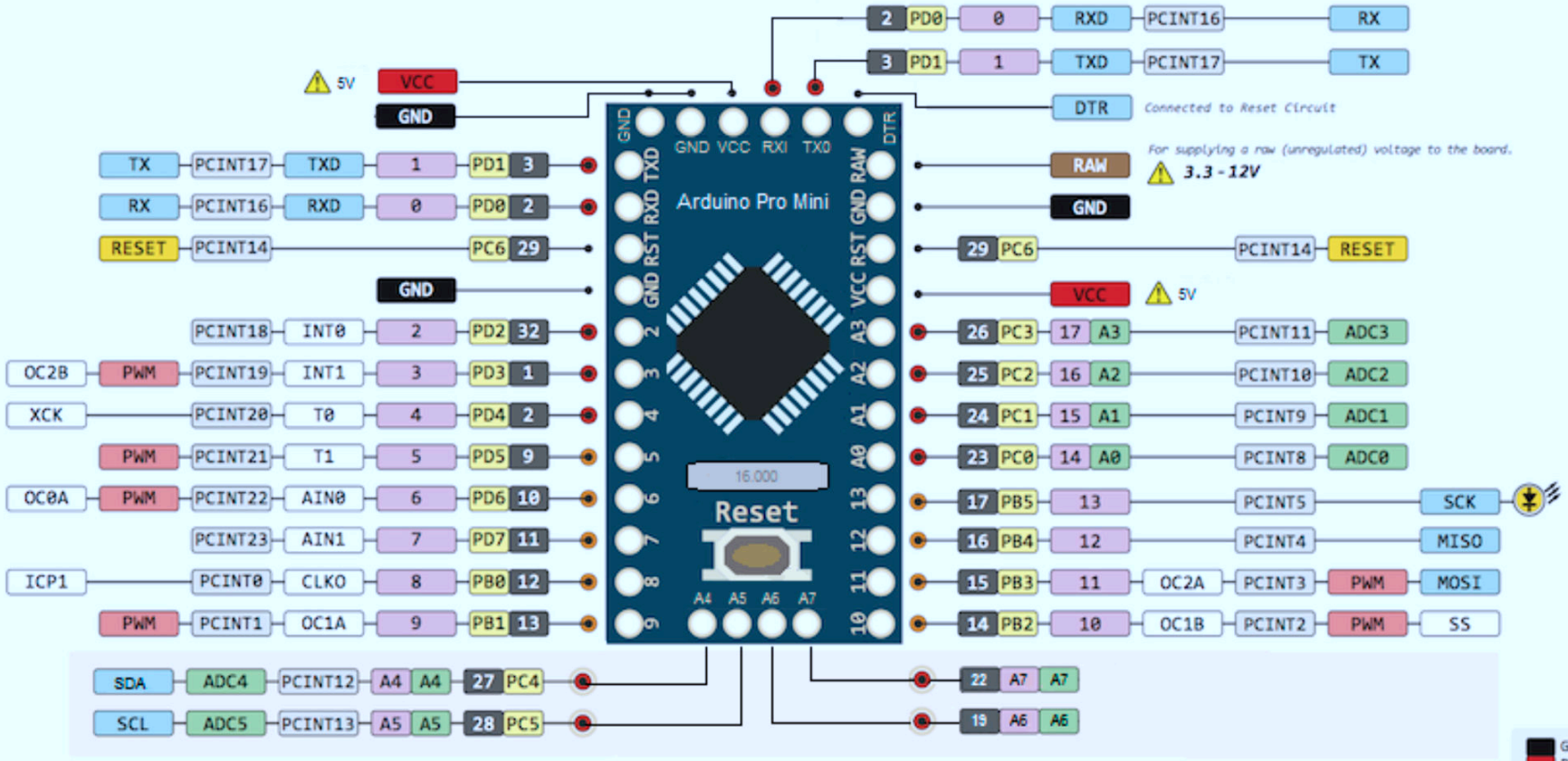
RoboCraft.ru

THE
UNOFFICIAL

ARDUINO ProMini

PINOUT DIAGRAM



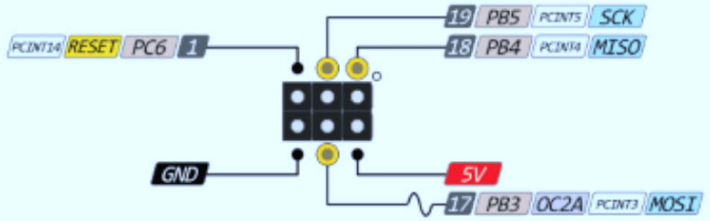


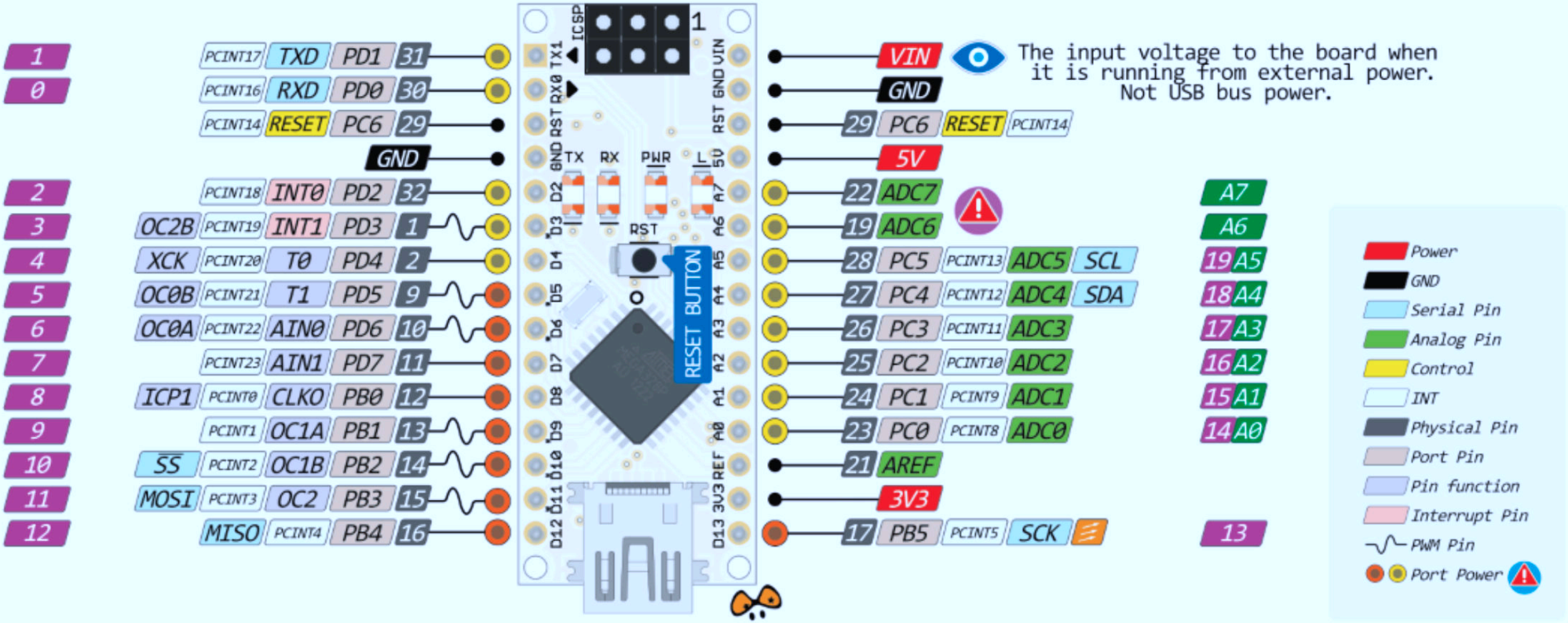
Absolute max per pin 40mA
reccomended 20mA

Absolute max 200mA
for entire package

- GND
- Power
- Control
- Physical Pin
- Port Pin
- Pin Function
- Digital Pin
- Analog Related Pin
- PWM Pin
- Serial Pin
- IDE
- Source Total 150mA

NANO PINOUT





The input voltage to the board when it is running from external power. Not USB bus power.

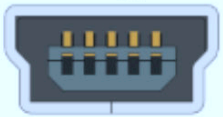
- Power
- GND
- Serial Pin
- Analog Pin
- Control
- INT
- Physical Pin
- Port Pin
- Pin function
- Interrupt Pin
- PWM Pin
- Port Power

The power sum for each pin's group should not exceed 100mA

Absolute MAX per pin 40mA
recommended 20mA

Absolute MAX 200mA
for entire package

USB JACK
Mini Type B

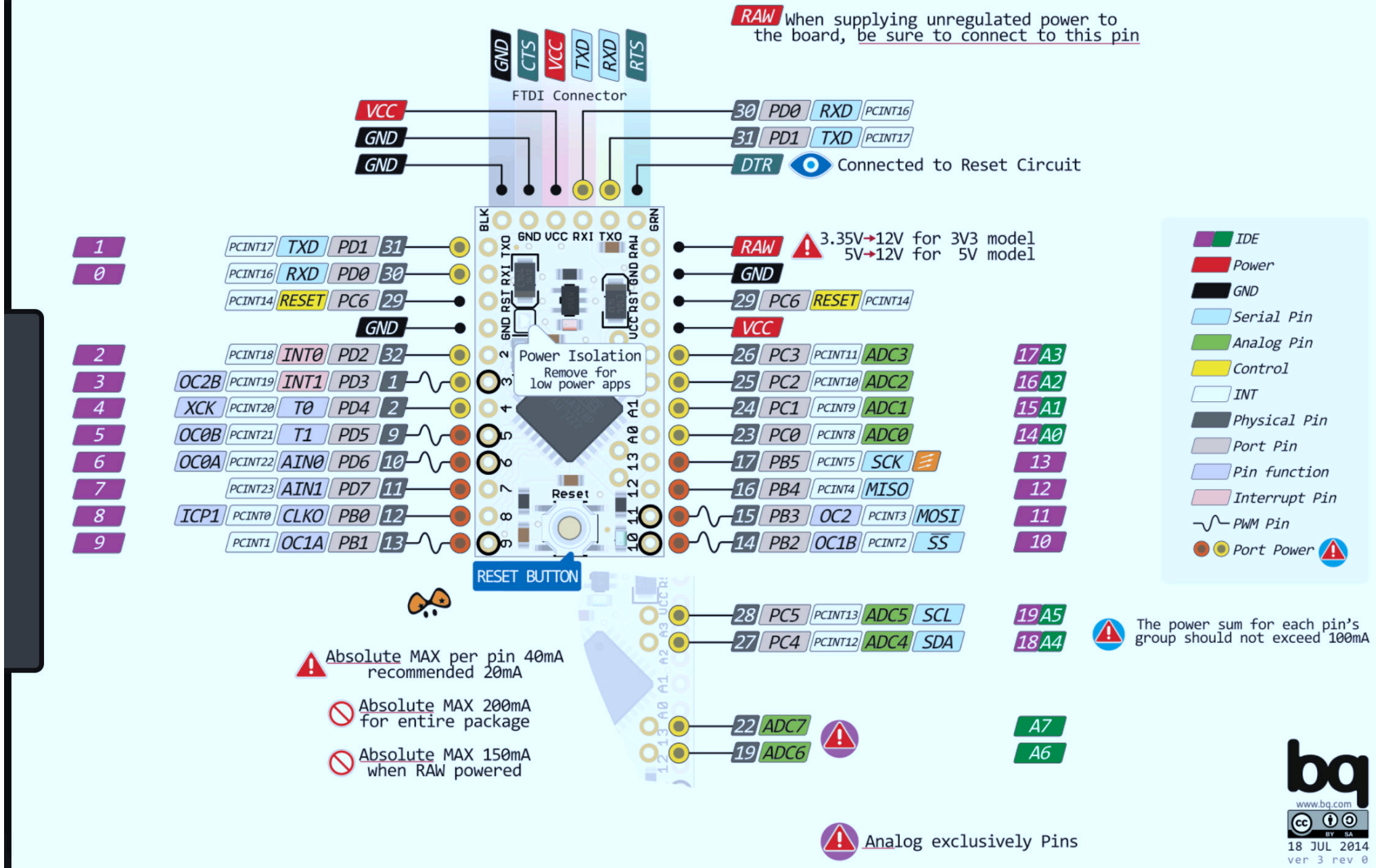


Analog exclusively Pins

bq
www.bq.com

19 AUG 2014
ver 3 rev 1

PRO MINI PINOUT

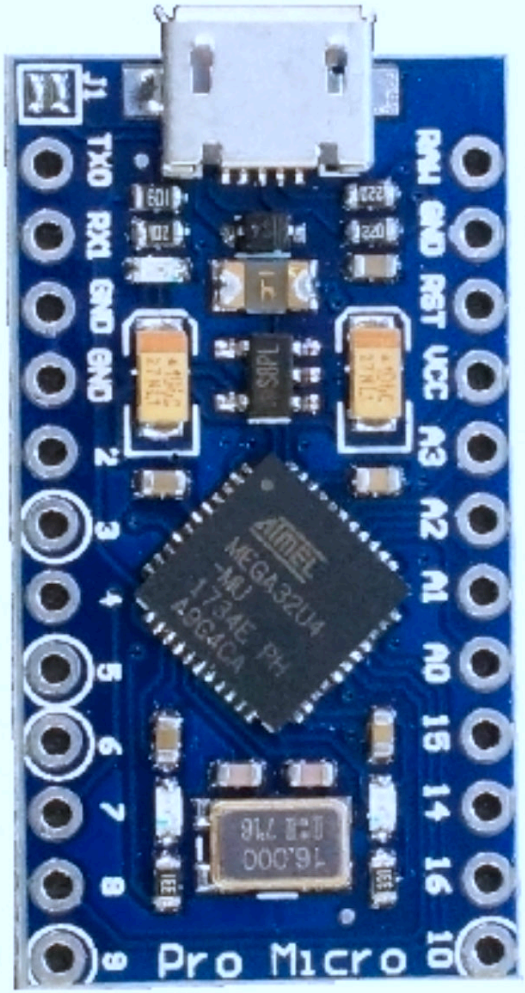


18 JUL 2014
ver 3 rev 0

Proto Supplies

Pro Micro Pinout

USB Micro-B Port
To Computer



Serial Port TXD / Ext Int 3 / Digital Pin 1 **(1)**

Serial Port RXD / Ext Int 2 / Digital Pin 0 **(0)**

Ground

Ground

(I2C) SDA / Ext Int 1 / Digital Pin 2 **(2)**

(I2C) SCL / Ext Int 0 / PWM / Digital Pin 3 **(3)**

Analog Pin 6 / Digital Pin 4 **(4)**

PWM / Digital Pin 5 **(5)**

PWM / Analog 7 / Digital Pin 6 **(6)**

Ext Int 6 / Digital Pin 7 **(7)**

Analog Pin 8 / Digital Pin 8 **(8)**

PWM / Analog Pin 9 / Digital Pin 9 / PWM **(9)**

7-12V Power Input

Ground

Reset Input (Ground to reset board)

5V Output or Input

(A3) Analog Pin 3 / Digital Pin 21

(A2) Analog Pin 2 / Digital Pin 20

(A1) Analog Pin 1 / Digital Pin 19

(A0) Analog Pin 0 / Digital Pin 18

(15) Digital Pin 15 / (SPI) SCK

(14) Digital Pin 14 / PWM / (SPI) MISO

(16) Digital Pin 16 / (SPI) MOSI

(10) Digital Pin 10 / Analog Pin 10 / PWM

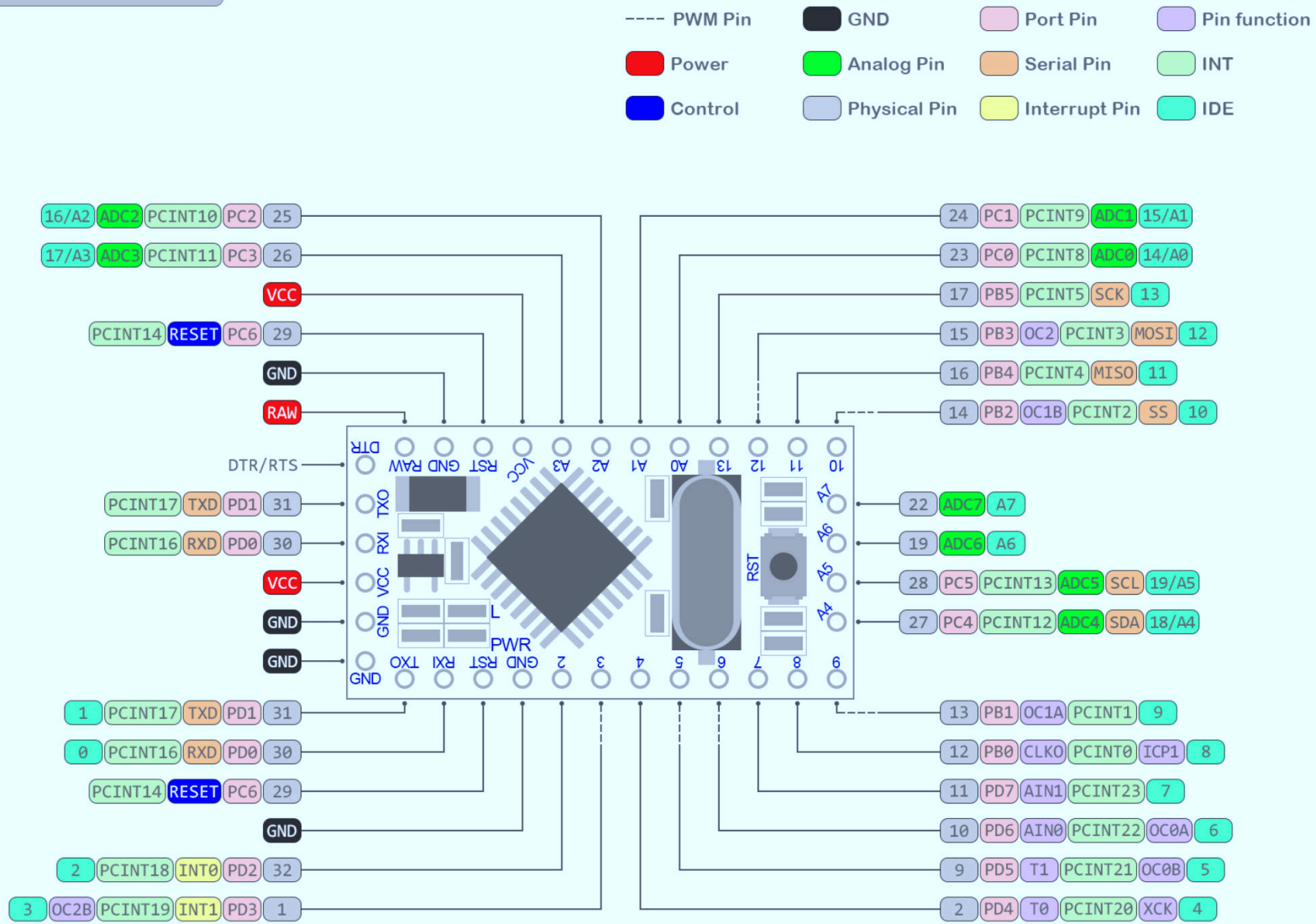
Red numbers in paranthesis are the name to use when referencing that pin.
Analog pins are references as A0 thru A3 even when using as digital I/O

RobotDyn®

www.robotdyn.com

PINOUT DIAGRAM

ProMini ATmega328P



jQuery API/1.2 http://jquery.com

SELECTORS

#id, tag, .class, *
elm1, elm2, elmN
ancestor descendant
parent > child
parent/child
prev + next
prev ~ siblings

:first
:last
:not(selector)
:even
:odd
:eq(index)
:lt(index)
:gt(index)

:contains(text)
:empty
:has(selector)
:parent

E[@attr]
E[@attr=val]
E[@attr^=val] (begins)
E[@attr$=val] (ends)
E[@attr*=val] (contains)
E[@attr=val][@attr=val] (both)

:nth-child(index)
:first-child
:last-child
:only-child

:enabled
:disabled
:checked
:selected

:hidden
:visible
:header
:animated

:input
:text
:password
:radio
:checkbox
:submit
:image
:reset
:button
:file
:hidden

CSS

.css(name, value)
.css(properties)

.height(value)
.width(value)

.addClass(class)
.removeClass(class)
.toggleClass(class)
.offset( )

ATTRIBUTES

.attr(name)
.attr(properties)
.attr(key, value)
.attr(key, function)
.removeAttr(name)

HTML

.html( )
.html(value)
.text( ), .text(value)
.val(value)

EVENTS

HANDLERS
.bind(type, data, fn)
.one(type, data, fn)
.trigger(type, data)
.triggerHandler(type, data)
.unbind(type, data)

MOUSE
.mousedown(fn)
.mousemove(fn)
.mouseout(fn)
.mouseover(fn)
.mouseup(fn)

WINDOW
.load(fn)
.scroll(fn)
.resize(fn)

ERROR
.error( )
.error(fn)

KEYBOARD
.keydown( )
.keydown(fn)
.keypress( )
.keypress(fn)
.keyup( )
.keyup(fn)

PAGE
.ready(fn)

INTERACTION
.hover(fnIN, fnOUT)
.toggle(fnIN, fnOUT)

.blur( )	.blur(fn)
.change( )	.change(fn)
.click( )	.click(fn)
.dblclick( )	.dblclick(fn)
.focus( )	.focus(fn)
.select( )	.select(fn)
.submit( )	.submit(fn)
.unload( )	.unload(fn)
.unblur( )	.unblur(fn)

CORE UI EFFECTS

SHOW / HIDE
.show( )
.show(speed, callback)
.hide( )
.hide(speed, callback)
.toggle( )

SLIDE (speed, callback)
.slideDown(s, c)
.slideUp(s, c)
.slideToggle(s, c)

FADE
.fadeIn(speed, callback)
.fadeOut(speed, callback)
.fadeTo(speed, opacity, callback)

ANIMATE
.stop( )
.queue( ),
.queue(callback),
.queue(queue)
.dequeue( )
.animate(params, duration, easing, callback)
.animate(params, options)

TRAVERSING

FILTER
.hasClass(class)
.filter(expr)
.filter(fn)
.is(expr)
.map(callback)
.not(expr)
.slice(start, end)

ACCESS
.each(callback)
.size( )
.length
.get( )
.get(index)
.index(subject)

FIND (expr)
.add(e)
.children(e), .siblings(e)
.contents( )
.find(e)
.next(e), .nextAll(expr)
.parent(e), .parents(e)
.prev(e), prevAll(e)

CHAIN
.andSelf( )
.end( )

MANIPULATING

INSIDE (content)
.append(c)
.appendTo(c)
.prepend(c)
.prependTo(c)

OUTSIDE (content)
.after(c)
.before(c)
.insertAfter(c)
.insertBefore(c)

AROUND
.wrap(html)
.wrap(element)
.wrapAll(html)
.wrapAll(element)
.wrapInner(html)
.wrapInner(element)

REPLACE
.replaceWith(c)
.replaceAll(selector)

CLEAR
.empty( )
.remove(expression)

CLONE
.clone( ) .clone(true)

AJAX

Request (url, data, callback)
$.ajax(options)
.load(u, d, c)
$.get(u, d, c)
$.getJSON(u, d, c)
$.getScript(u, c)
$.post(u, d, c)
.loadIfModified(u, d, c)

Event Handler (callback)
.ajaxComplete(c) .ajaxError(c)
.ajaxSend(c) .ajaxStart(c)
.ajaxStop(c) .ajaxSuccess(c)

Serialize
.serialize( )
.serializeArray( )
.ajaxSetup(options)

USER AGENT

$.browser, $.browser.version
$.boxModel

JavaScript

$.extend(obj1,...objN)
$.grep(array, callback, invert)
$.map(array, callback)
$.unique (array)
$.trim(string)
$.merge(1st, 2nd)

COLORCHARGE

http://labs.colorcharge.com/jquery/
Revision: 2007-September-26

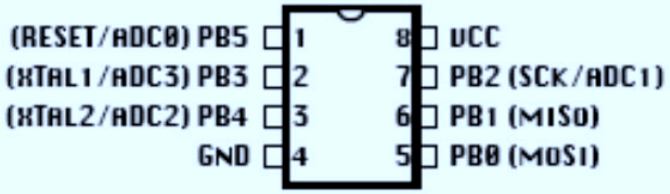
EXTEND

$.fn.extend(obj)
$.extend(obj)
$.noConflict( )

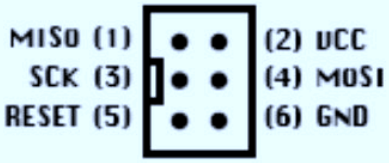
$( );

$(expression, context), .$(html)
$(elements), $(callback)

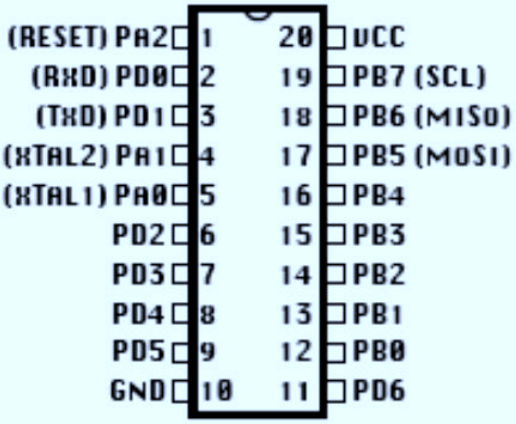
ATtiny25/45/85



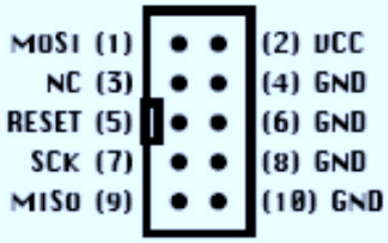
ISP Header 6-pin



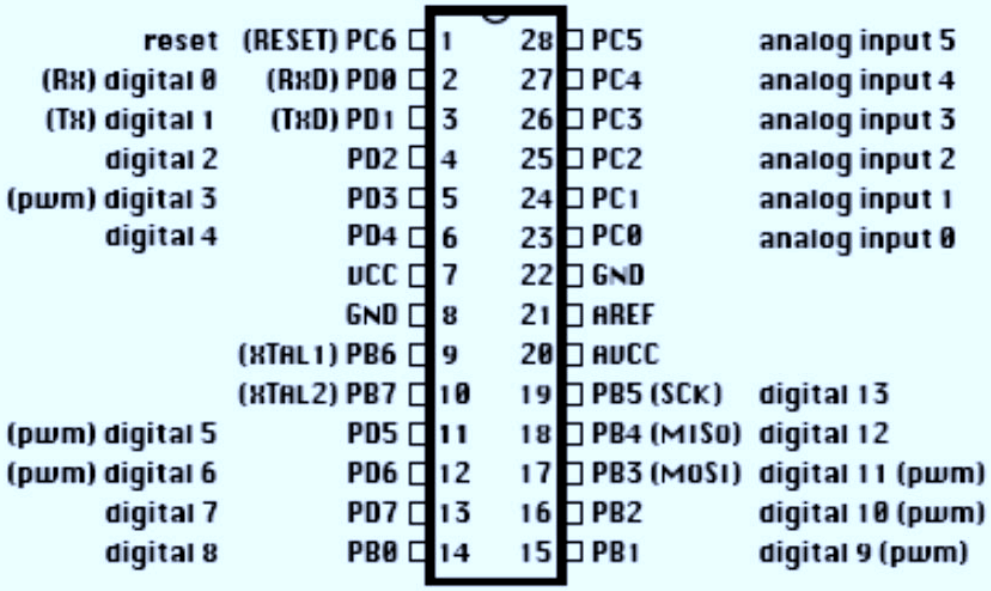
ATtiny2313



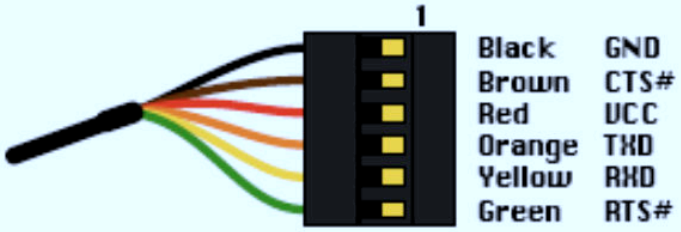
ISP Header 10-pin



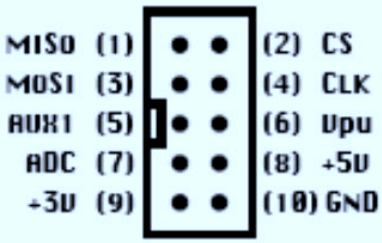
ATmega48/88/168/328 Arduino



FTDI Cable



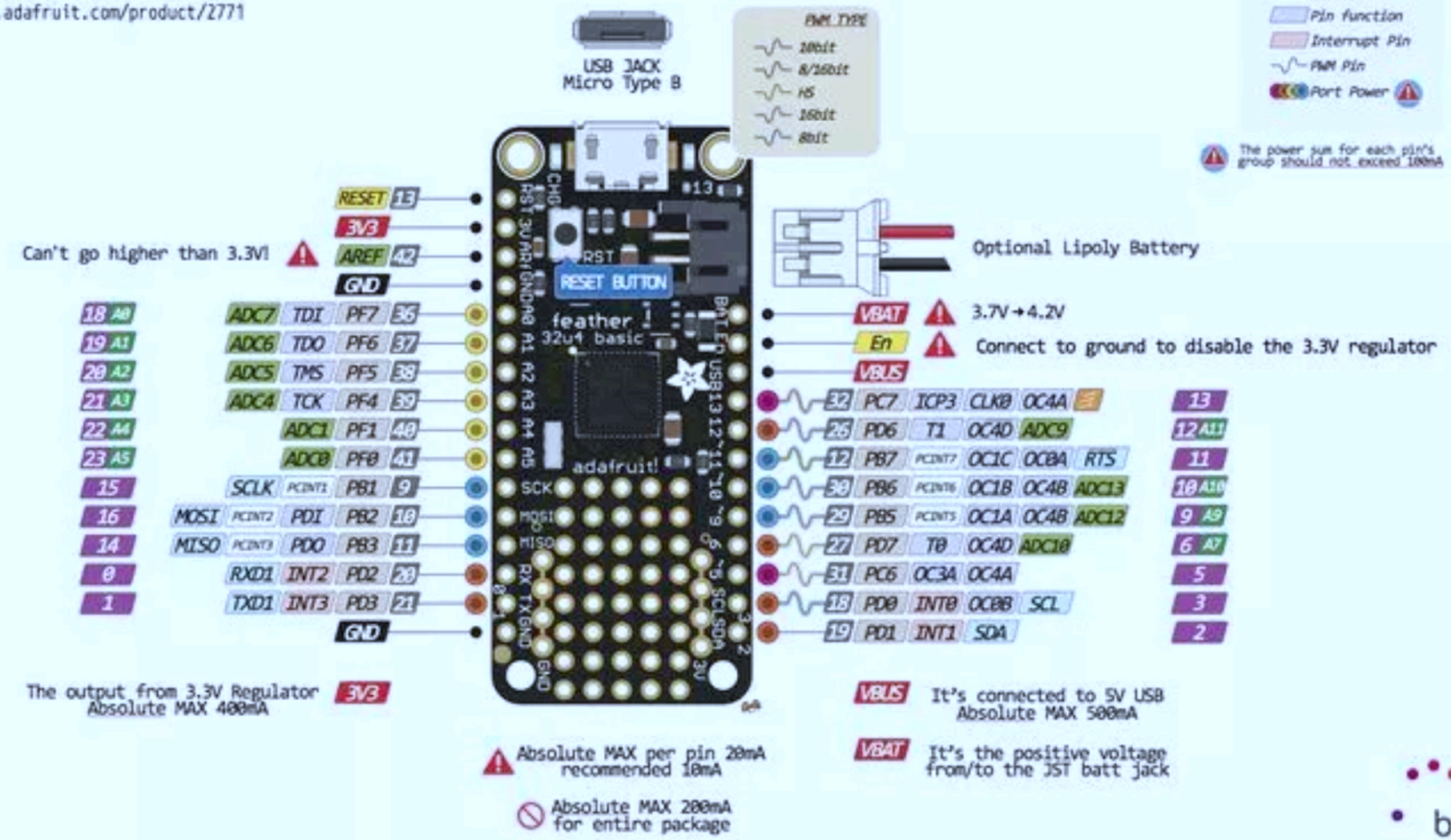
BusPirate I/O Header



feather

32u4 Basic Proto

https://www.adafruit.com/product/2771



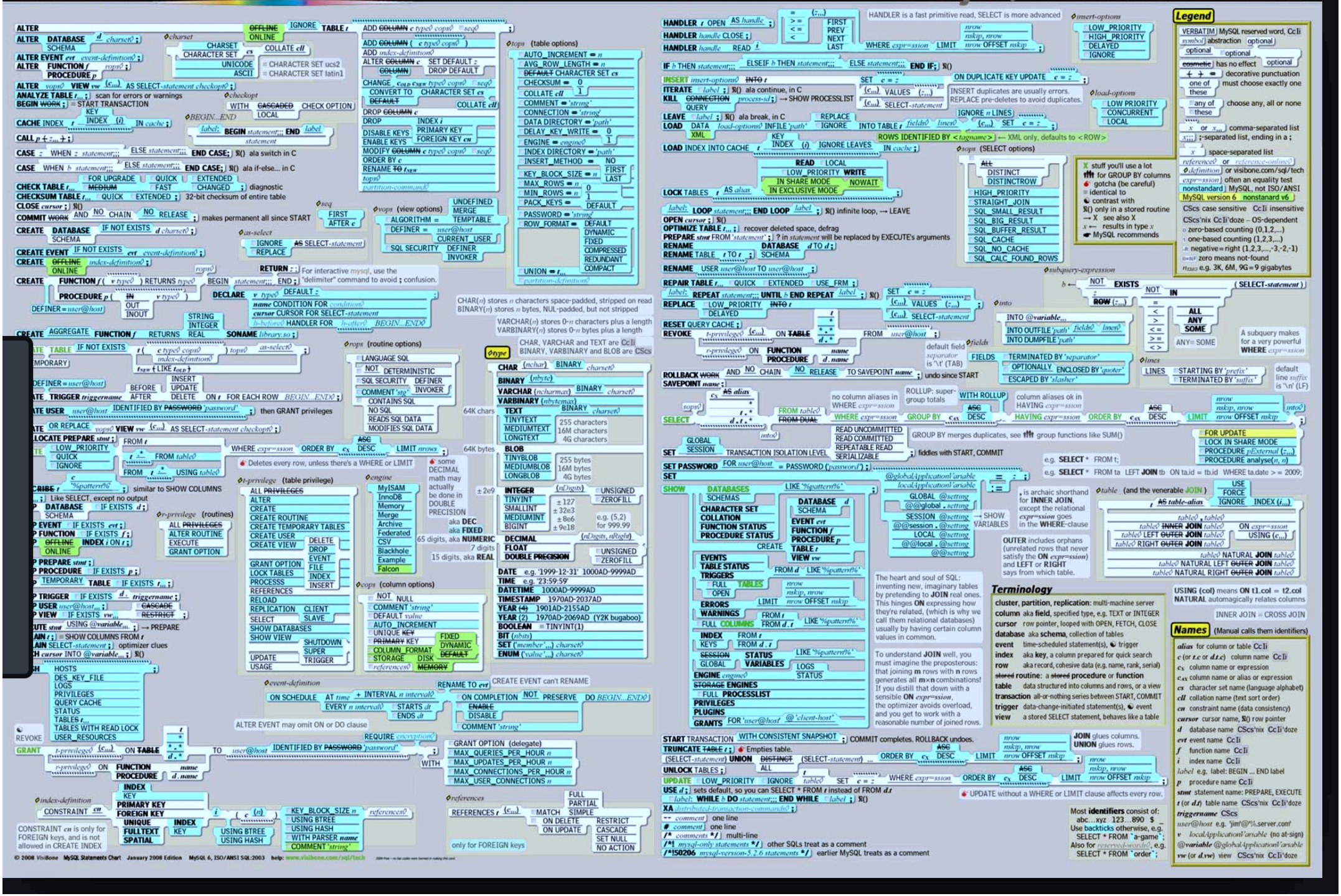
adafruit

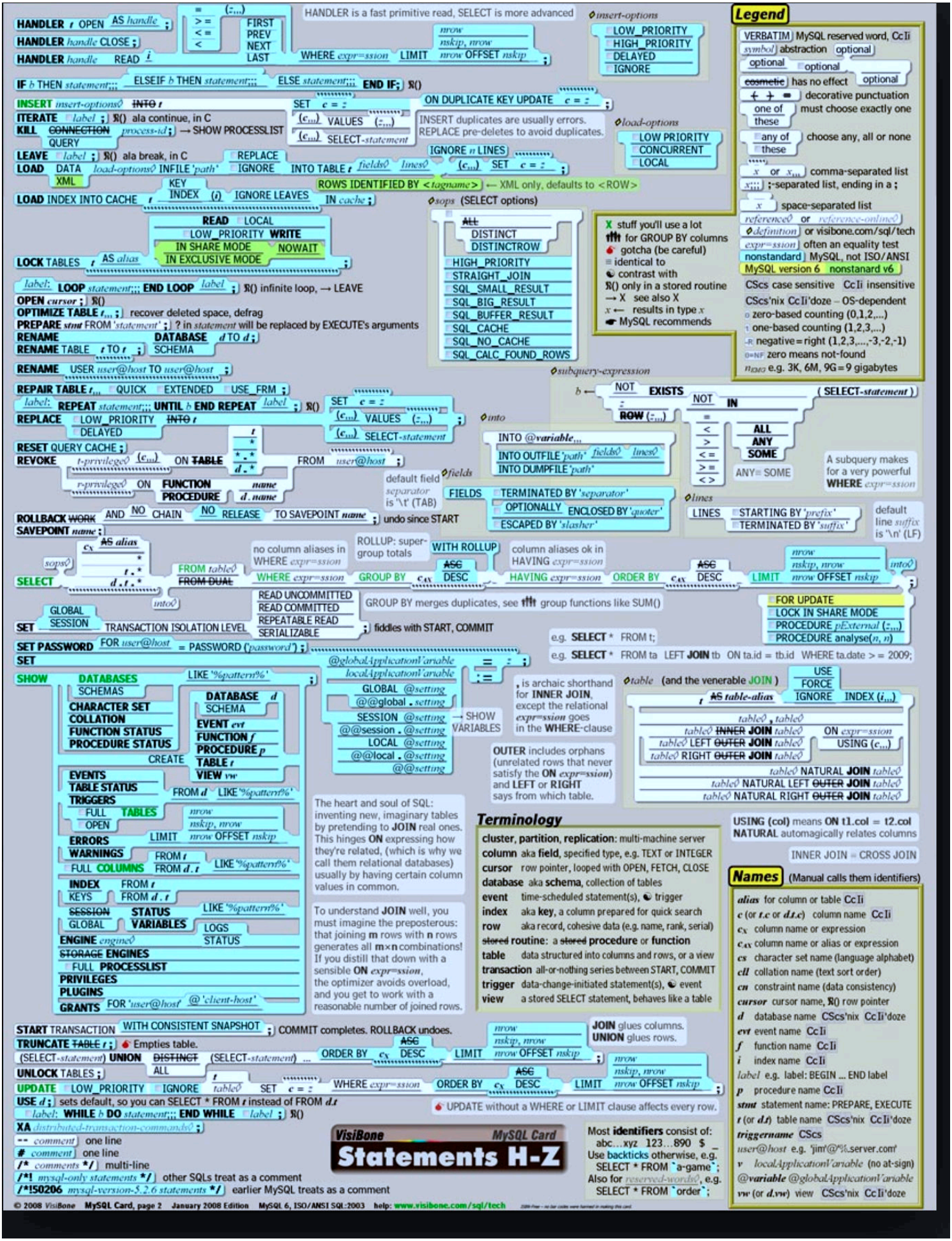
bq

26 JAN 2016

ver 1 rev 1

VisiBone — MySQL Statements Chart

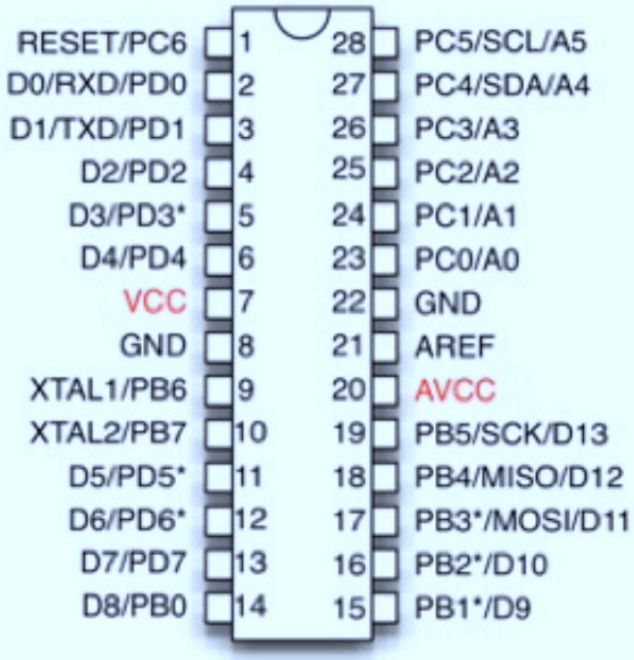



HANDLER t OPEN AS handle ;
HANDLER handle CLOSE ;
HANDLER handle READ i = >= <= < (c,...) FIRST PREV NEXT LAST WHERE expr=ssion LIMIT nrow nskip, nrow nrow OFFSET nskip ;
HANDLER is a fast primitive read, SELECT is more advanced
IF b THEN statement;;; ELSEIF b THEN statement;;; ELSE statement;;; END IF; ℜ()
INSERT insert-options◊ INTO t SET c = z (c,...) VALUES (z,...) (c,...) SELECT-statement ON DUPLICATE KEY UPDATE c = z ;
INSERT duplicates are usually errors. REPLACE pre-deletes to avoid duplicates.
ITERATE label ; ℜ() ala continue, in C
KILL CONNECTION QUERY process-id ; → SHOW PROCESSLIST
LEAVE label ; ℜ() ala break, in C
LOAD DATA XML load-options◊ INFILE 'path' REPLACE IGNORE INTO TABLE t fields◊ lines◊ IGNORE n LINES (c,...) SET c = z ;
LOAD INDEX INTO CACHE t KEY INDEX (i) IGNORE LEAVES IN cache ;
ROWS IDENTIFIED BY <tagname> ← XML only, defaults to <ROW>
LOCK TABLES t AS alias READ LOCAL LOW_PRIORITY WRITE IN SHARE MODE NOWAIT IN EXCLUSIVE MODE ;
label: LOOP statement;;; END LOOP label ; ℜ() infinite loop, → LEAVE
OPEN cursor ; ℜ()
OPTIMIZE TABLE t,... ; recover deleted space, defrag
PREPARE stmt FROM 'statement' ; ? in statement will be replaced by EXECUTE's arguments
RENAME DATABASE d TO d ;
RENAME TABLE t TO t ; SCHEMA
RENAME USER user@host TO user@host ;
REPAIR TABLE t,... QUICK EXTENDED USE_FRM ;
label: REPEAT statement;;; UNTIL b END REPEAT label ; ℜ()
REPLACE LOW_PRIORITY INTO t DELAYED SET c = z (c,...) VALUES (z,...) (c,...) SELECT-statement ;
RESET QUERY CACHE ;
REVOKE t-privilege◊ (c,...) ON TABLE t * *.* d.* FROM user@host ;
r-privilege◊ ON FUNCTION name PROCEDURE d.name
ROLLBACK WORK AND NO CHAIN NO RELEASE TO SAVEPOINT name ; undo since START
SAVEPOINT name ;
SELECT sops◊ cx AS alias * t.* d.t.* FROM table◊ FROM DUAL WHERE expr=ssion GROUP BY cAX ASC DESC WITH ROLLUP HAVING expr=ssion ORDER BY cAX ASC DESC LIMIT nrow nskip, nrow nrow OFFSET nskip into◊ ;
no column aliases in WHERE expr=ssion
ROLLUP: super-group totals
column aliases ok in HAVING expr=ssion
GROUP BY merges duplicates, see ☂ group functions like SUM()
SET GLOBAL SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED READ COMMITTED REPEATABLE READ SERIALIZABLE ; fiddles with START, COMMIT
SET PASSWORD FOR user@host = PASSWORD ('password') ;
SET @globalApplicationVariable localApplicationVariable = := ;
SHOW DATABASES SCHEMAS LIKE '%pattern%' ;
CHARACTER SET
COLLATION
FUNCTION STATUS
PROCEDURE STATUS
CREATE DATABASE d SCHEMA
EVENT evt
FUNCTION f
PROCEDURE p
TABLE t
VIEW vw
EVENTS
TABLE STATUS
TRIGGERS
FULL TABLES FROM d LIKE '%pattern%'
OPEN nrow nskip, nrow LIMIT nrow OFFSET nskip
ERRORS
WARNINGS
FULL COLUMNS FROM t FROM d.t LIKE '%pattern%'
INDEX FROM t
KEYS FROM d.t
SESSION STATUS LIKE '%pattern%'
GLOBAL VARIABLES
LOGS
STATUS
ENGINE engine◊
STORAGE ENGINES
FULL PROCESSLIST
PRIVILEGES
PLUGINS
GRANTS FOR 'user@host' @'client-host'
@globalApplicationVariable
localApplicationVariable
GLOBAL @setting
@@global.setting
SESSION @setting
@@session.@setting
LOCAL @setting
@@local.@setting
@@setting
→ SHOW VARIABLES
, is archaic shorthand for INNER JOIN, except the relational expr=ssion goes in the WHERE-clause
OUTER includes orphans (unrelated rows that never satisfy the ON expr=ssion) and LEFT or RIGHT says from which table.
The heart and soul of SQL: inventing new, imaginary tables by pretending to JOIN real ones. This hinges ON expressing how they're related, (which is why we call them relational databases) usually by having certain column values in common.
To understand JOIN well, you must imagine the preposterous: that joining m rows with n rows generates all m×n combinations! If you distill that down with a sensible ON expr=ssion, the optimizer avoids overload, and you get to work with a reasonable number of joined rows.
START TRANSACTION WITH CONSISTENT SNAPSHOT ; COMMIT completes. ROLLBACK undoes.
TRUNCATE TABLE t ; Empties table.
(SELECT-statement) UNION DISTINCT ALL (SELECT-statement) ... ORDER BY cx ASC DESC LIMIT nrow nskip, nrow nrow OFFSET nskip ;
UNLOCK TABLES ;
UPDATE LOW_PRIORITY IGNORE t table◊ SET c = z WHERE expr=ssion ORDER BY cx ASC DESC LIMIT nrow nskip, nrow nrow OFFSET nskip ;
USE d ; sets default, so you can SELECT * FROM t instead of FROM d.t
label: WHILE b DO statement;;; END WHILE label ; ℜ()
XA distributed-transaction-commands◊ ;
-- comment one line
comment one line
/* comments */ multi-line
/*! mysql-only statements */ other SQLs treat as a comment
/*!50206 mysql-version-5.2.6 statements */ earlier MySQL treats as a comment
UPDATE without a WHERE or LIMIT clause affects every row.
VisiBone MySQL Card
Statements H-Z
Most identifiers consist of: abc...xyz 123...890 $ _
Use backticks otherwise, e.g. SELECT * FROM `a-game`;
Also for reserved-words◊, e.g. SELECT * FROM `order`;
insert-options◊
LOW_PRIORITY
HIGH_PRIORITY
DELAYED
IGNORE
load-options◊
LOW PRIORITY
CONCURRENT
LOCAL
sops◊ (SELECT options)
ALL
DISTINCT
DISTINCTROW
HIGH_PRIORITY
STRAIGHT_JOIN
SQL_SMALL_RESULT
SQL_BIG_RESULT
SQL_BUFFER_RESULT
SQL_CACHE
SQL_NO_CACHE
SQL_CALC_FOUND_ROWS
X stuff you'll use a lot
☂ for GROUP BY columns
gotcha (be careful)
≡ identical to
☯ contrast with
ℜ() only in a stored routine
→ X see also X
x ← results in type x
MySQL recommends
subquery-expression◊
b ← NOT EXISTS (SELECT-statement)
z ROW (z,...)
NOT IN
= < > <= >= <> ALL ANY SOME
ANY≡SOME
A subquery makes for a very powerful WHERE expr=ssion
into◊
INTO @variable,...
INTO OUTFILE 'path' fields◊ lines◊
INTO DUMPFILE 'path'
fields◊
default field separator is '\t' (TAB)
FIELDS TERMINATED BY 'separator'
OPTIONALLY ENCLOSED BY 'quoter'
ESCAPED BY 'slasher'
lines◊
LINES STARTING BY 'prefix'
TERMINATED BY 'suffix'
default line suffix is '\n' (LF)
FOR UPDATE
LOCK IN SHARE MODE
PROCEDURE pExternal (z,...)
PROCEDURE analyse(n, n)
e.g. SELECT * FROM t;
e.g. SELECT * FROM ta LEFT JOIN tb ON ta.id = tb.id WHERE ta.date >= 2009;
table◊ (and the venerable JOIN)
t AS table-alias USE FORCE IGNORE INDEX (i,...)
table◊ , table◊
table◊ INNER JOIN table◊ ON expr=ssion
table◊ LEFT OUTER JOIN table◊ USING (c,...)
table◊ RIGHT OUTER JOIN table◊
table◊ NATURAL JOIN table◊
table◊ NATURAL LEFT OUTER JOIN table◊
table◊ NATURAL RIGHT OUTER JOIN table◊
USING (col) means ON t1.col = t2.col
NATURAL automagically relates columns
INNER JOIN ≡ CROSS JOIN
JOIN glues columns.
UNION glues rows.
Terminology
cluster, partition, replication: multi-machine server
column aka field, specified type, e.g. TEXT or INTEGER
cursor row pointer, looped with OPEN, FETCH, CLOSE
database aka schema, collection of tables
event time-scheduled statement(s), ☯ trigger
index aka key, a column prepared for quick search
row aka record, cohesive data (e.g. name, rank, serial)
stored routine: a stored procedure or function
table data structured into columns and rows, or a view
transaction all-or-nothing series between START, COMMIT
trigger data-change-initiated statement(s), ☯ event
view a stored SELECT statement, behaves like a table
Legend
VERBATIM MySQL reserved word, CcIi
symbol abstraction optional
optional optional
cosmetic has no effect optional
+ + = decorative punctuation
one of these must choose exactly one
any of these choose any, all or none
x or x,... comma-separated list
x;;; ;-separated list, ending in a ;
x space-separated list
reference◊ or reference-online◊
definition◊ or visibone.com/sql/tech
expr=ssion often an equality test
nonstandard MySQL, not ISO/ANSI
MySQL version 6 nonstanard v6
CScs case sensitive CcIi insensitive
CScs'nix CcIi'doze – OS-dependent
0 zero-based counting (0,1,2,...)
1 one-based counting (1,2,3,...)
-R negative=right (1,2,3,...,-3,-2,-1)
0=NF zero means not-found
nMG e.g. 3K, 6M, 9G=9 gigabytes
Names (Manual calls them identifiers)
alias for column or table CcIi
c (or t.c or d.t.c) column name CcIi
cx column name or expression
cAX column name or alias or expression
cs character set name (language alphabet)
cll collation name (text sort order)
cn constraint name (data consistency)
cursor cursor name, ℜ() row pointer
d database name CScs'nix CcIi'doze
evt event name CcIi
f function name CcIi
i index name CcIi
label e.g. label: BEGIN ... END label
p procedure name CcIi
stmt statement name: PREPARE, EXECUTE
t (or d.t) table name CScs'nix CcIi'doze
triggername CScs
user@host e.g. 'jim'@'%.server.com'
v localApplicationVariable (no at-sign)
@variable @globalApplicationVariable
vw (or d.vw) view CScs'nix CcIi'doze

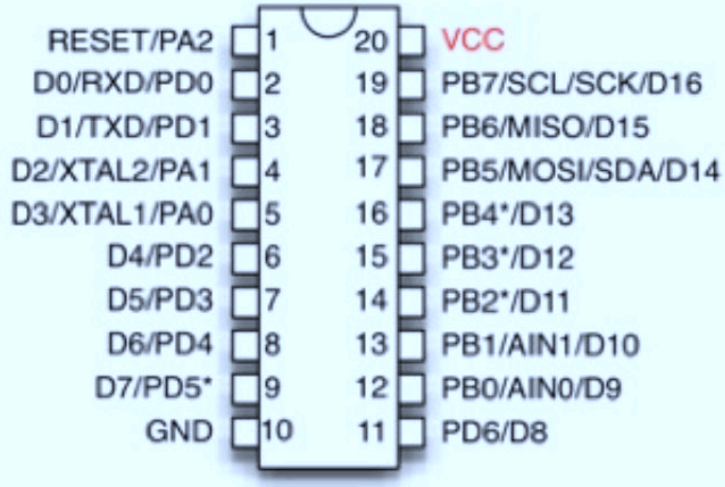
赤 akafugu.jp

Microcontroller Reference Sheet v1.2

ATMega8/48/88/168/328/Arduino

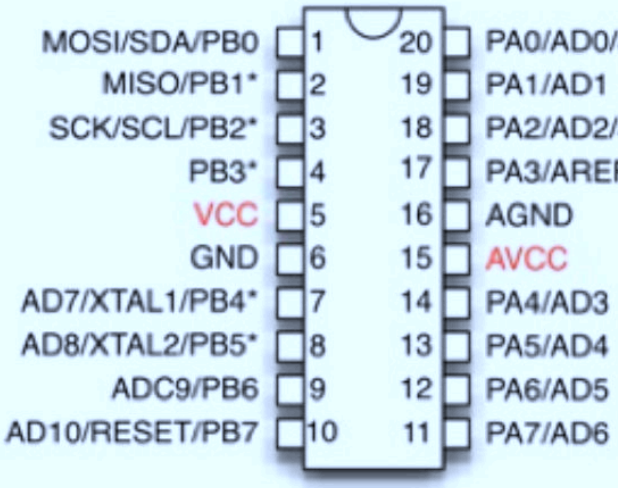


ATTiny2313/4313



ATTiny24/44/84

ATTiny26/261/461/861

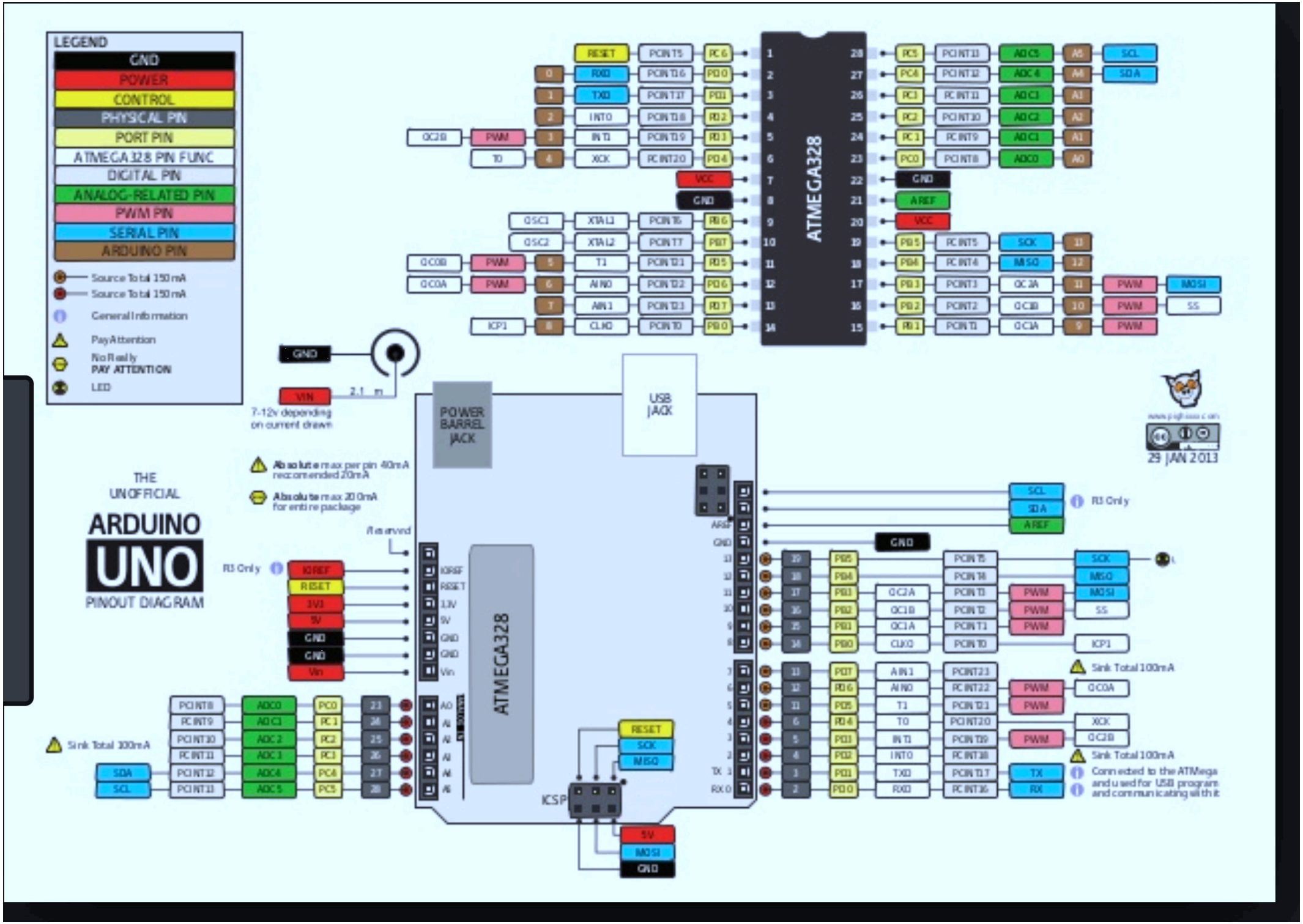


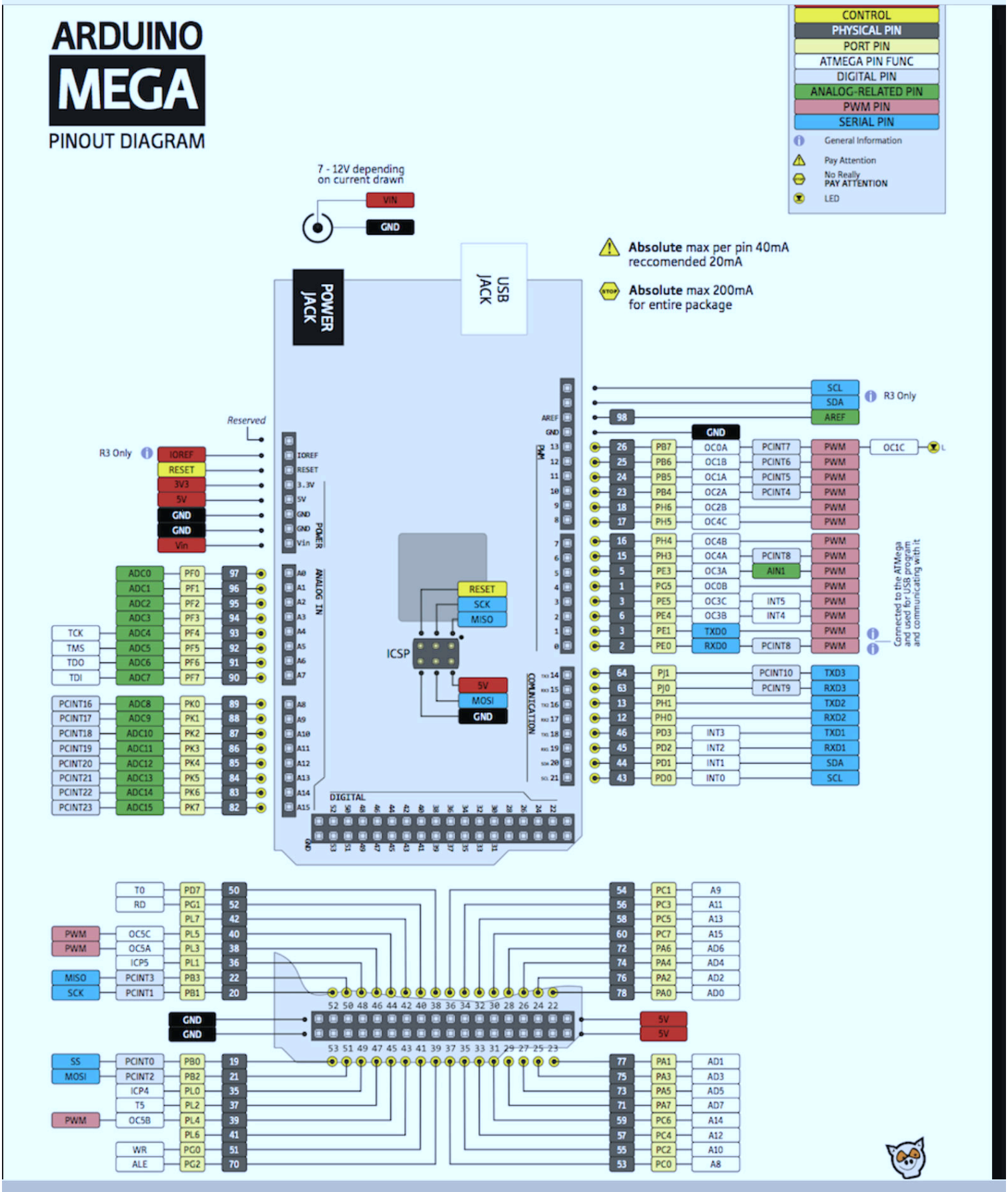
ATTiny25/45/85/13

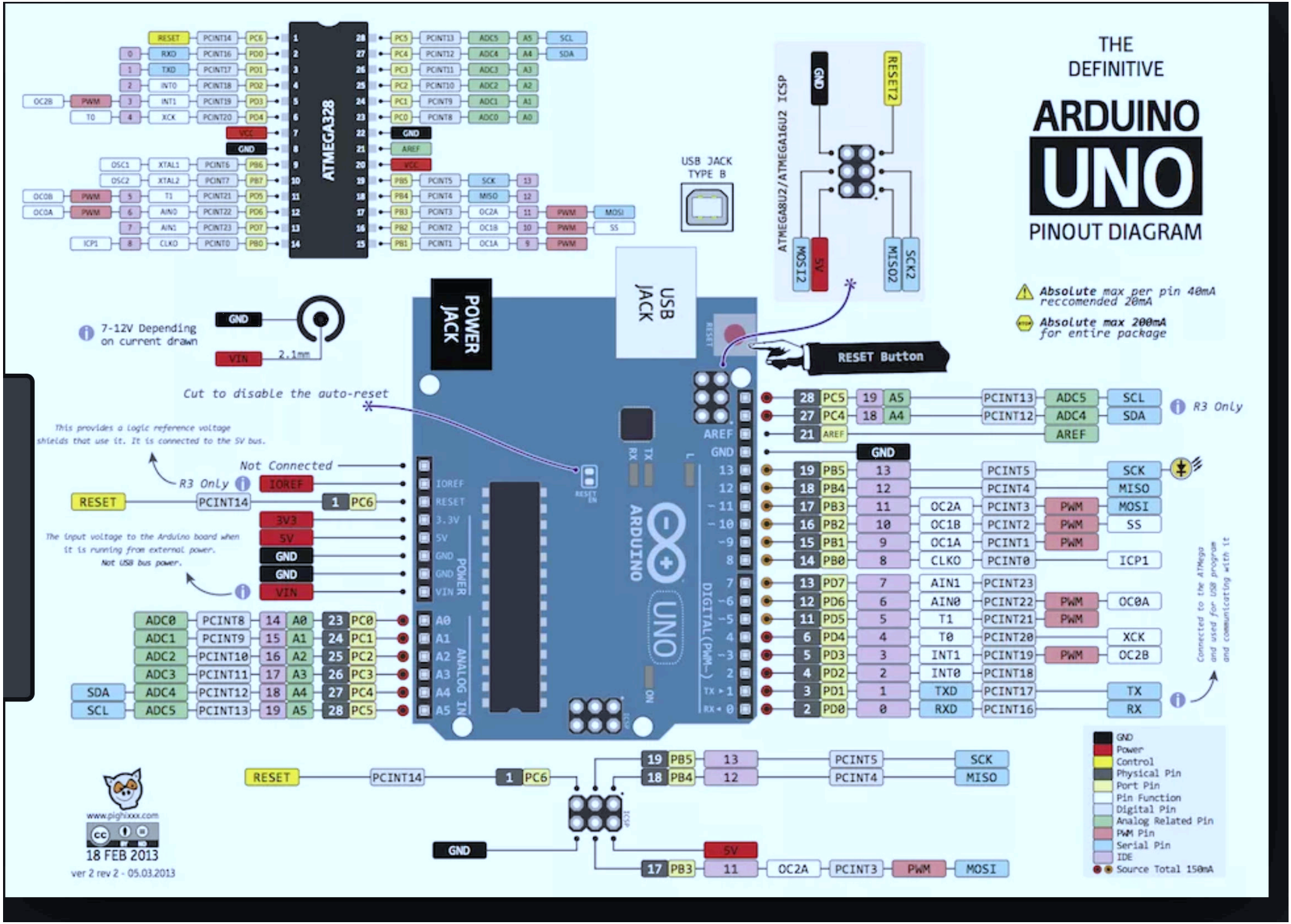
ISP Header

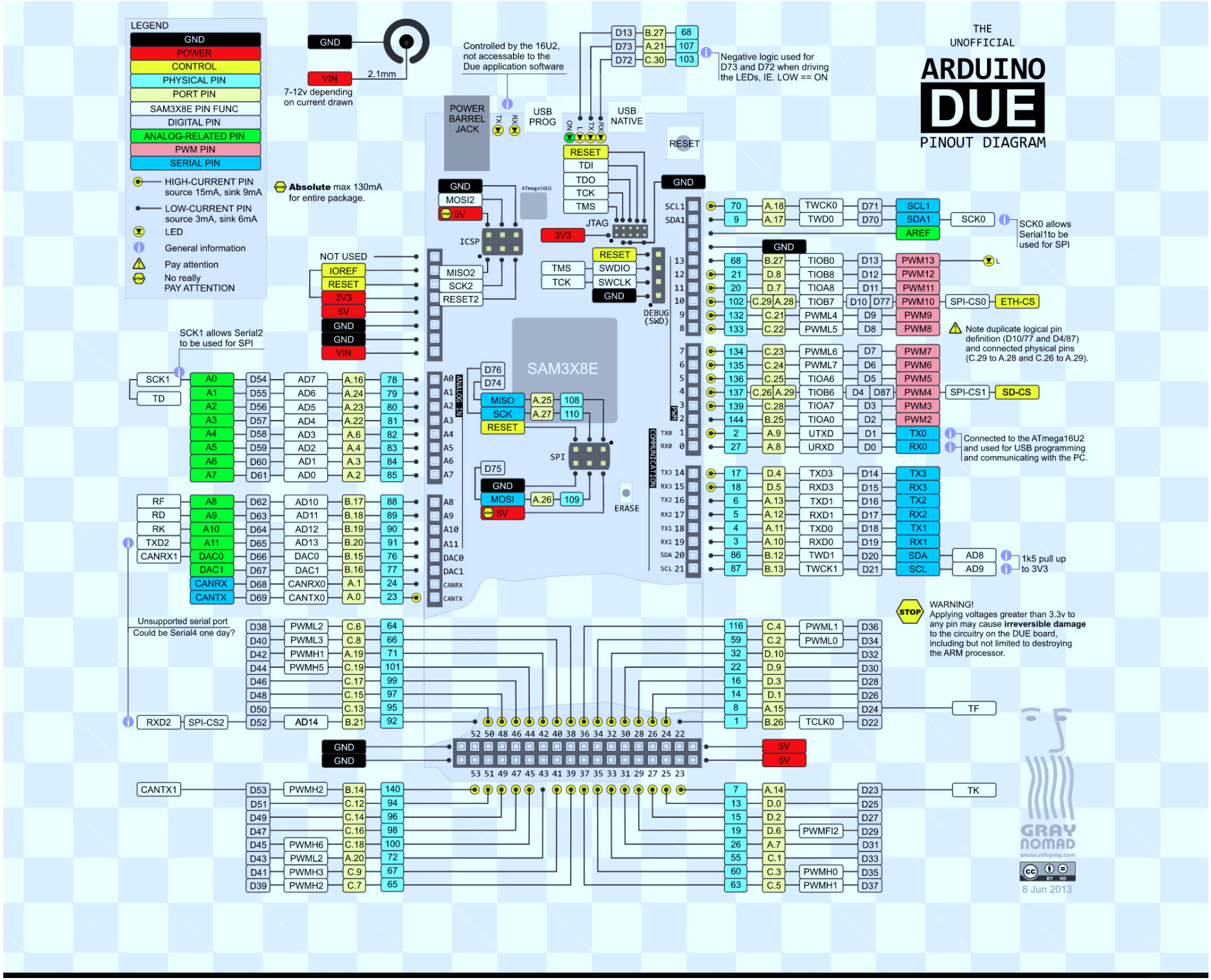
NOTES:
- Arduino pins for ATTinyX313/X5/X4 are from the arduino-tiny project
- PWM pins are marked with (*)

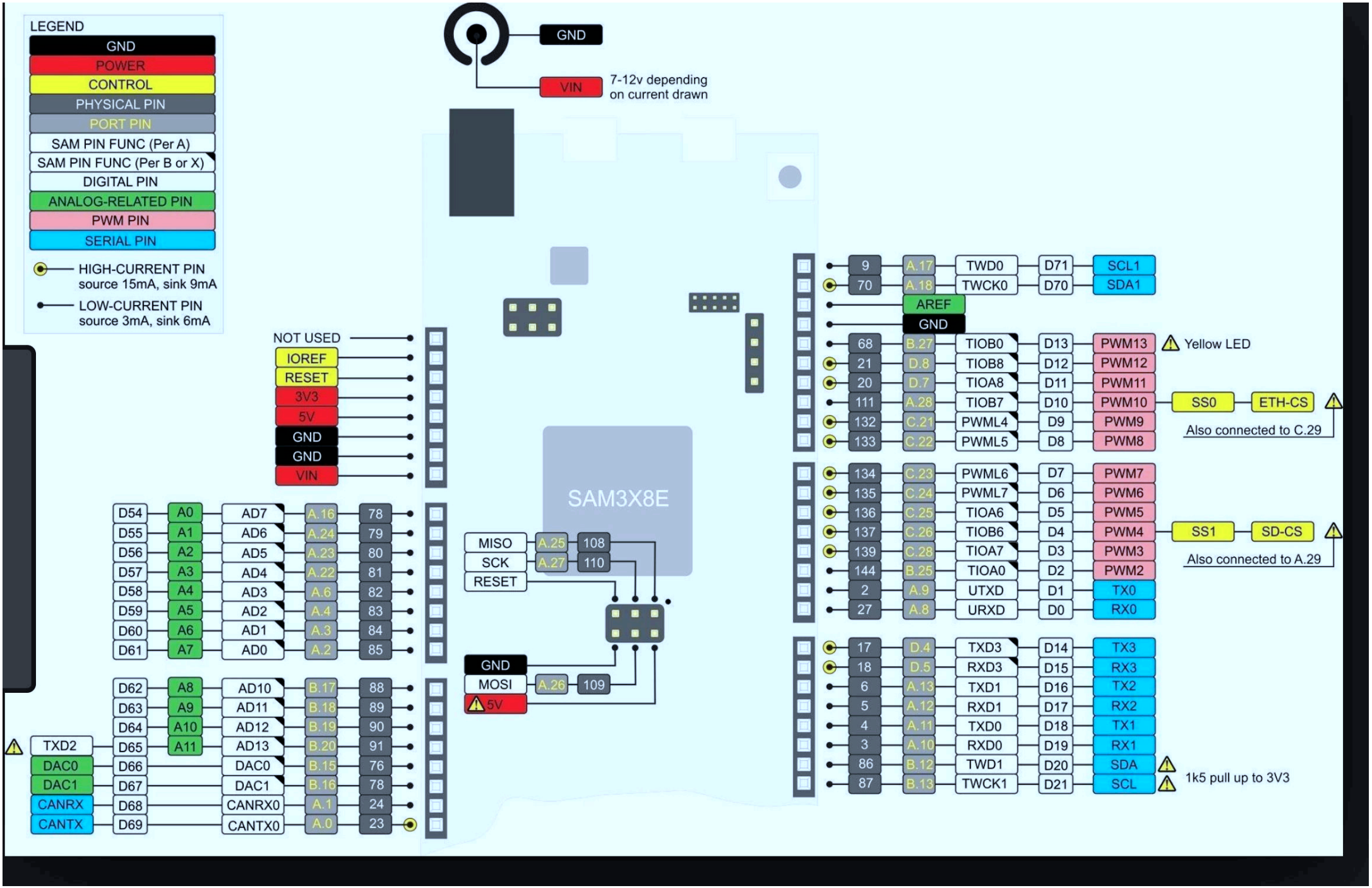
*ATTiny13 has PWM only on PB0 and PB1
* No AREF on ATTiny13
* PB3 and PB4 share the same timer


LEGEND
GND
POWER
CONTROL
PHYSICAL PIN
PORT PIN
ATMEGA328 PIN FUNC
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
ARDUINO PIN
Source Total 150mA
Source Total 150mA
General Information
Pay Attention
No Really
PAY ATTENTION
LED
ATMEGA328
POWER BARREL JACK
USB JACK
7-12v depending on current drawn
Absolute max per pin 40mA recommended 20mA
Absolute max 200mA for entire package
THE UNOFFICIAL
ARDUINO UNO
PINOUT DIAGRAM
29 JAN 2013
R3 Only
Sink Total 100mA
Connected to the ATMega and used for USB program and communicating with it
ICSP


ARDUINO
MEGA
PINOUT DIAGRAM
CONTROL
PHYSICAL PIN
PORT PIN
ATMEGA PIN FUNC
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
General Information
Pay Attention
No Really
PAY ATTENTION
LED
7 - 12V depending
on current drawn
VIN
GND
POWER
JACK
USB
JACK
Absolute max per pin 40mA
reccomended 20mA
Absolute max 200mA
for entire package
Reserved
R3 Only
IOREF
RESET
3V3
5V
GND
GND
Vin
POWER
ANALOG IN
ADC0
ADC1
ADC2
ADC3
ADC4
ADC5
ADC6
ADC7
TCK
TMS
TDO
TDI
PF0
PF1
PF2
PF3
PF4
PF5
PF6
PF7
97
96
95
94
93
92
91
90
PCINT16
PCINT17
PCINT18
PCINT19
PCINT20
PCINT21
PCINT22
PCINT23
ADC8
ADC9
ADC10
ADC11
ADC12
ADC13
ADC14
ADC15
PK0
PK1
PK2
PK3
PK4
PK5
PK6
PK7
89
88
87
86
85
84
83
82
ICSP
RESET
SCK
MISO
5V
MOSI
GND
AREF
GND
PWM
COMMUNICATION
DIGITAL
SCL
SDA
AREF
R3 Only
98
GND
26
25
24
23
18
17
PB7
PB6
PB5
PB4
PH6
PH5
OC0A
OC1B
OC1A
OC2A
OC2B
OC4C
PCINT7
PCINT6
PCINT5
PCINT4
PWM
OC1C
16
15
5
1
3
6
3
2
PH4
PH3
PE3
PG5
PE5
PE4
PE1
PE0
OC4B
OC4A
OC3A
OC0B
OC3C
OC3B
TXD0
RXD0
PCINT8
AIN1
INT5
INT4
PCINT8
Connected to the ATMega
and used for USB program
and communicating with it
64
63
13
12
46
45
44
43
PJ1
PJ0
PH1
PH0
PD3
PD2
PD1
PD0
PCINT10
PCINT9
INT3
INT2
INT1
INT0
TXD3
RXD3
TXD2
RXD2
TXD1
RXD1
SDA
SCL
T0
RD
PWM
PWM
MISO
SCK
PD7
PG1
PL7
PL5
PL3
PL1
PB3
PB1
50
52
42
40
38
36
22
20
OC5C
OC5A
ICP5
PCINT3
PCINT1
52 50 48 46 44 42 40 38 36 34 32 30 28 26 24 22
GND
GND
5V
5V
53 51 49 47 45 43 41 39 37 35 33 31 29 27 25 23
SS
MOSI
PWM
WR
ALE
PCINT0
PCINT2
ICP4
T5
OC5B
PB0
PB2
PL0
PL2
PL4
PL6
PG0
PG2
19
21
35
37
39
41
51
70
54
56
58
60
72
74
76
78
PC1
PC3
PC5
PC7
PA6
PA4
PA2
PA0
A9
A11
A13
A15
AD6
AD4
AD2
AD0
77
75
73
71
59
57
55
53
PA1
PA3
PA5
PA7
PC6
PC4
PC2
PC0
AD1
AD3
AD5
AD7
A14
A12
A10
A8


THE DEFINITIVE
ARDUINO UNO
PINOUT DIAGRAM
ATMEGA328
ATMEGA8U2/ATMEGA16U2 ICSP
USB JACK TYPE B
POWER JACK
USB JACK
RESET Button
Absolute max per pin 40mA
reccomended 20mA
Absolute max 200mA
for entire package
7-12V Depending on current drawn
2.1mm
Cut to disable the auto-reset
This provides a logic reference voltage shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
The input voltage to the Arduino board when it is running from external power.
Not USB bus power.
Connected to the ATMega and used for USB program and communicating with it
R3 Only
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013


THE UNOFFICIAL
ARDUINO DUE
PINOUT DIAGRAM
LEGEND
GND
POWER
CONTROL
PHYSICAL PIN
PORT PIN
SAM3X8E PIN FUNC
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
HIGH-CURRENT PIN source 15mA, sink 9mA
LOW-CURRENT PIN source 3mA, sink 6mA
LED
General information
Pay attention
No really PAY ATTENTION
Absolute max 130mA for entire package.
GND
VIN
2.1mm
7-12v depending on current drawn
Controlled by the 16U2, not accessable to the Due application software
Negative logic used for D73 and D72 when driving the LEDs, IE. LOW == ON
POWER BARREL JACK
USB PROG
USB NATIVE
RESET
TDI
TDO
TCK
TMS
JTAG
ICSP
MISO2
SCK2
RESET2
3V3
SWDIO
SWCLK
DEBUG (SWD)
NOT USED
IOREF
SCK1 allows Serial2 to be used for SPI
SCK1
TD
SAM3X8E
MISO
SCK
SPI
ERASE
SCK0 allows Serial1to be used for SPI
SCK0
AREF
SPI-CS0
ETH-CS
SPI-CS1
SD-CS
Note duplicate logical pin definition (D10/77 and D4/87) and connected physical pins (C.29 to A.28 and C.26 to A.29).
Connected to the ATmega16U2 and used for USB programming and communicating with the PC.
1k5 pull up to 3V3
Unsupported serial port
Could be Serial4 one day?
RXD2
SPI-CS2
TF
TK
CANTX1
WARNING!
Applying voltages greater than 3.3v to any pin may cause irreversible damage to the circuitry on the DUE board, including but not limited to destroying the ARM processor.
GRAY NOMAD
www.robgray.com
8 Jun 2013


LEGEND
GND
POWER
CONTROL
PHYSICAL PIN
PORT PIN
SAM PIN FUNC (Per A)
SAM PIN FUNC (Per B or X)
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
HIGH-CURRENT PIN source 15mA, sink 9mA
LOW-CURRENT PIN source 3mA, sink 6mA
GND
VIN
7-12v depending on current drawn
NOT USED
IOREF
RESET
3V3
5V
GND
GND
VIN
SAM3X8E
MISO
SCK
RESET
GND
MOSI
5V
A.25
A.27
A.26
108
110
109
9 A.17 TWD0 D71 SCL1
70 A.18 TWCK0 D70 SDA1
AREF
GND
68 B.27 TIOB0 D13 PWM13
21 D.8 TIOB8 D12 PWM12
20 D.7 TIOA8 D11 PWM11
111 A.28 TIOB7 D10 PWM10
132 C.21 PWML4 D9 PWM9
133 C.22 PWML5 D8 PWM8
Yellow LED
SS0
ETH-CS
Also connected to C.29
134 C.23 PWML6 D7 PWM7
135 C.24 PWML7 D6 PWM6
136 C.25 TIOA6 D5 PWM5
137 C.26 TIOB6 D4 PWM4
139 C.28 TIOA7 D3 PWM3
144 B.25 TIOA0 D2 PWM2
2 A.9 UTXD D1 TX0
27 A.8 URXD D0 RX0
SS1
SD-CS
Also connected to A.29
17 D.4 TXD3 D14 TX3
18 D.5 RXD3 D15 RX3
6 A.13 TXD1 D16 TX2
5 A.12 RXD1 D17 RX2
4 A.11 TXD0 D18 TX1
3 A.10 RXD0 D19 RX1
86 B.12 TWD1 D20 SDA
87 B.13 TWCK1 D21 SCL
1k5 pull up to 3V3
D54 A0 AD7 A.16 78
D55 A1 AD6 A.24 79
D56 A2 AD5 A.23 80
D57 A3 AD4 A.22 81
D58 A4 AD3 A.6 82
D59 A5 AD2 A.4 83
D60 A6 AD1 A.3 84
D61 A7 AD0 A.2 85
D62 A8 AD10 B.17 88
D63 A9 AD11 B.18 89
D64 A10 AD12 B.19 90
D65 A11 AD13 B.20 91
TXD2
DAC0 D66 DAC0 B.15 76
DAC1 D67 DAC1 B.16 78
CANRX D68 CANRX0 A.1 24
CANTX D69 CANTX0 A.0 23

Protect Your Boat With the Correct Size Wire, Fuse, and Fuse Holder

U.S. Coast Guard and other regulatory agencies require all circuits, except the starting circuit, to be protected with a circuit breaker or a fuse.

For 24V DC Systems divide distance by 2 or consult the Circuit Wizard at www.circuitwizard.bluesea.com

STEP 1 Choose the Correct Wire

Calculations are based on 105°C wire. For more detailed calculations, download the Circuit Wizard app or go to www.circuitwizard.bluesea.com

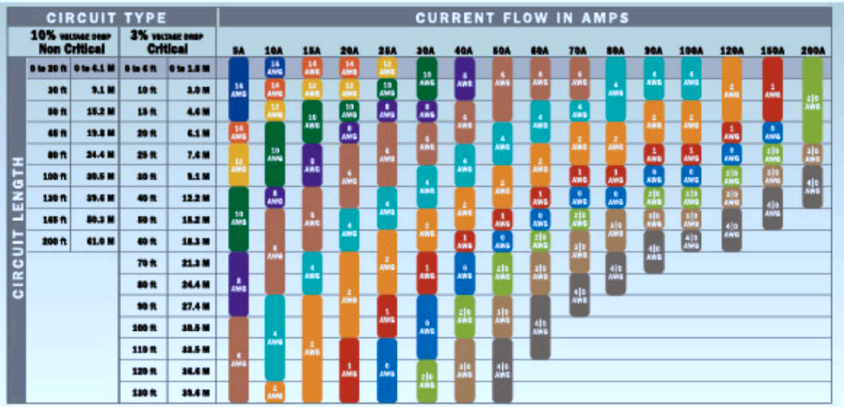
A Locate the **CURRENT FLOW IN AMPS** of your circuit along the top of the chart to the right.

B Select the **CIRCUIT TYPE.**
Non-critical circuits with 10% allowable voltage drop include: general lighting, windlasses, bait pumps, general appliances
Critical circuits with 3% allowable voltage drop include: panel main feeders, bilge blowers, electronics, navigation lights

C Find the **CIRCUIT LENGTH** along the left side of the chart.
The circuit length is the length of the negative wire added to the length of the positive wire.
Calculations are based on 105°C wire. For wire rated at 90°C or lower, or for wire that passes through an engine room, the first row of the chart, in gray, does not apply.

D Intersect the **CURRENT FLOW IN AMPS** with **CIRCUIT LENGTH** to identify the correct wire size.
Example: A windlass rated 80A is 25 ft. from the battery. The circuit length is the total length of the positive and negative wire added together, which in this example is 50 ft. The circuit type is 'non-critical', and the correct wire size is 4 AWG.

AWG WIRE SIZE CHART Circles indicate actual diameter of wire (not including insulation)



STEP 2 Choose the Correct Fuse and Fuse Amperage

Calculations are based on 105°C wire. For lower temperature rated wire, download the Circuit Wizard app or go to www.circuitwizard.bluesea.com

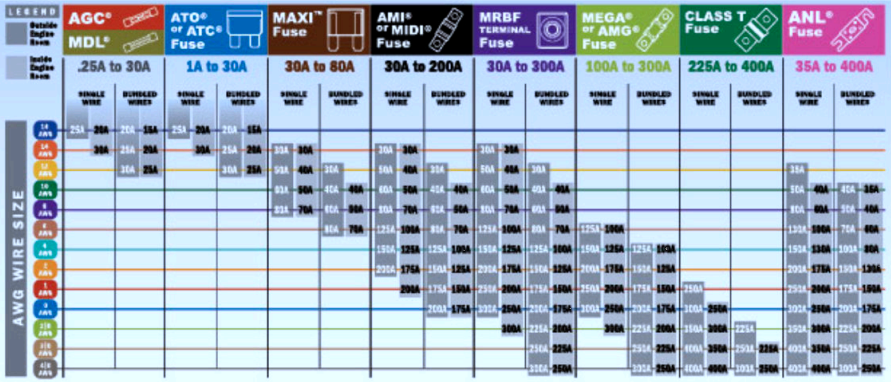
A Choose a fuse from the list on the top of the chart to the right by following along the line of the **AWG WIRE SIZE** determined from Step 1. Appropriate fuses will have a gray bar that intersects the line.

B The appropriate fuse amperage will be found in one of the four gray bars below the selected fuse type.
- **Single Wire, Outside Engine Room** = First column dark gray bar
- **Single Wire, Inside Engine Room** = First column light gray bar
- **Bundled Wire, Outside Engine Room** = Second column dark gray bar
- **Bundled Wire, Inside Engine Room** = Second column light gray bar

Example: For a 4 AWG single 105°C rated wire outside an engine room, the maximum fuse amperage is 150A.

Note:
Possible fuse amperages for a circuit can fall between a range of maximum and minimum fuse amperages. The procedure above calculates the maximum fuse amperage which reduces nuisance blows but may offer less protection than a lower amperage fuse. The minimum fuse amperage is calculated by multiplying the current flow in amps by 125%.

If the product instructions specify a fuse amperage, use that value if it is under the maximum amperage found in the above procedure. If the specified fuse amperage is over the maximum suggested, move down the column and choose the wire size that intersects with the specified fuse amperage.

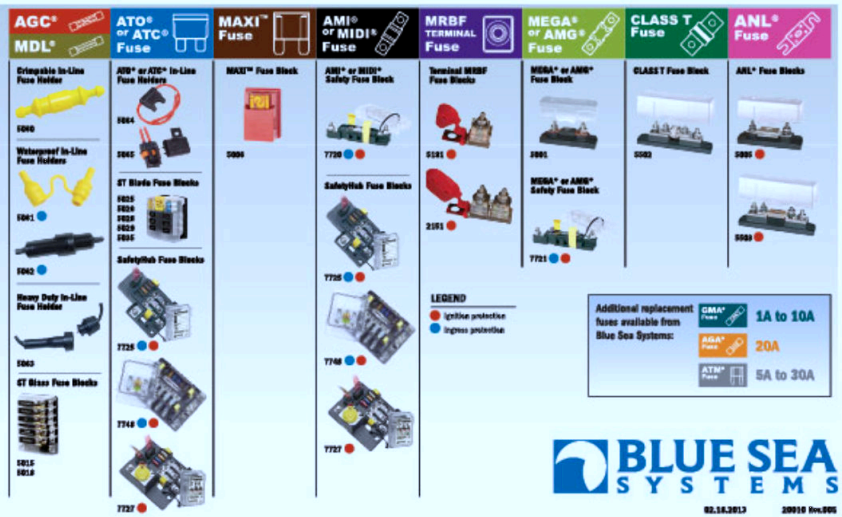


STEP 3 Choose a Fuse Holder

A Using the same colored headings as in the steps above, follow the columns down to find fuse holders or fuse blocks that meet your specific requirements.

B Consider environmental factors:
- Ignition protection is required where flammable vapors may accumulate.
  Example: Engine room and propane locker
  Consult American Boat and Yacht Council (ABYC) E-11.5.3 for Ignition Protection
  Ignition protection
- Ingress protection protects fuses from spray, washdown, and humidity.
  IP66-protected against powerful water jets
  Ingress protection

C Decide between an in-line fuse holder or a fuse block:
- In-line fuse holders are compact and hold a single low-amperage fuse.
- Fuse blocks mount to a solid surface and may hold a single fuse or multiple fuses.



Although this process uses information from ABYC E-11 to recommend wire size and circuit protection, it may not cover all of the unique characteristics that may exist on a boat. If you have specific questions about your installation please consult an ABYC certified installer.

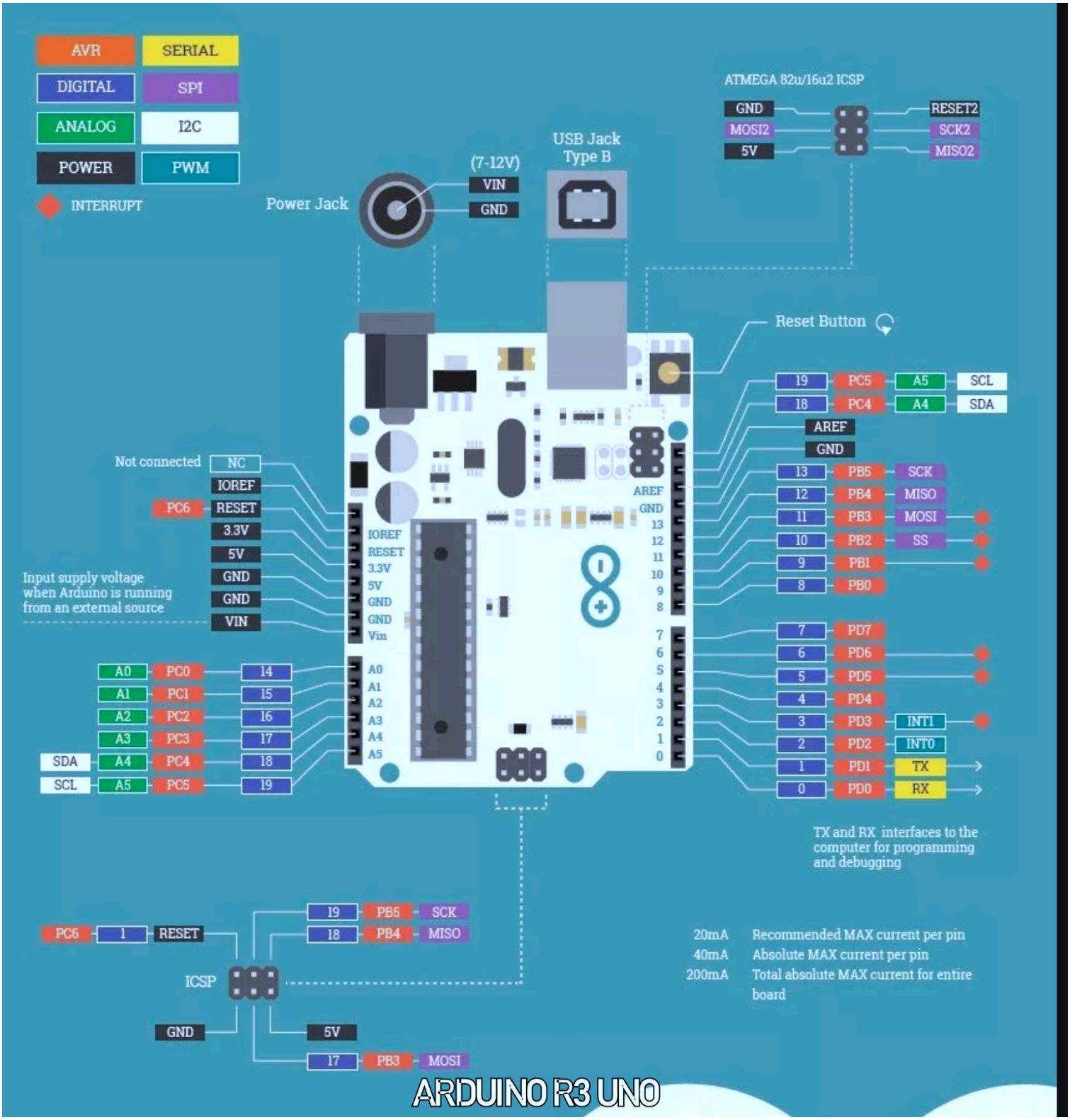
Blue Sea Systems
425 Sequoia Drive
Bellingham, WA 98226 USA
p 360.738.8230
p 800.222.7617 toll free
f 360.734.4195
www.bluesea.com
conductor@bluesea.com

Circuit Wizard App
Easily calculate the proper wire size, fuse, and circuit breaker from your mobile device or computer.
Available for Android and iPhone

Scan to download the app or go to www.circuitwizard.bluesea.com

BLUE SEA SYSTEMS

02.18.2013 20910 Rev.005



ARDUINO R3 UNO

LEGEND

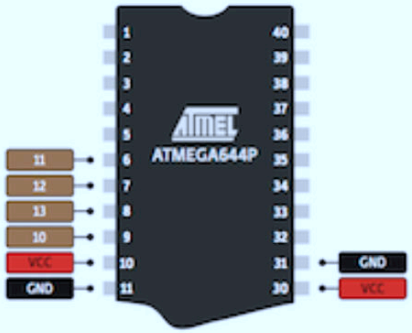
- GND
- POWER
- CONTROL
- PORT PIN
- ATMEGA PIN FUNC
- DIGITAL PIN
- ANALOG-RELATED PIN
- PWM PIN
- SERIAL PIN
- ARDUINO PIN

Using Arduino as ICSP Programmer for ATMEGA644P

THE UNOFFICIAL

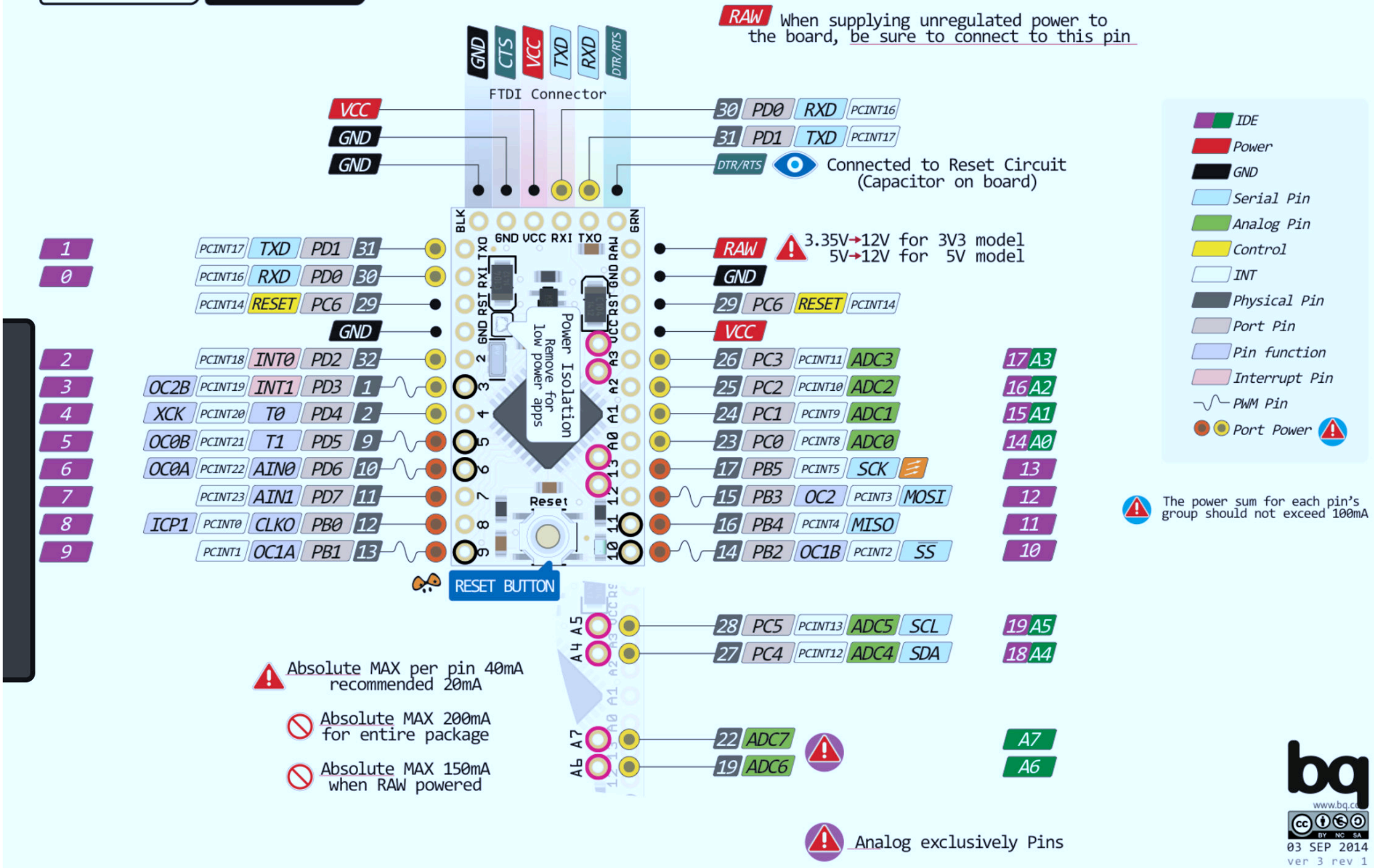
ARDUINO & ATMega644P

PINOUT DIAGRAM



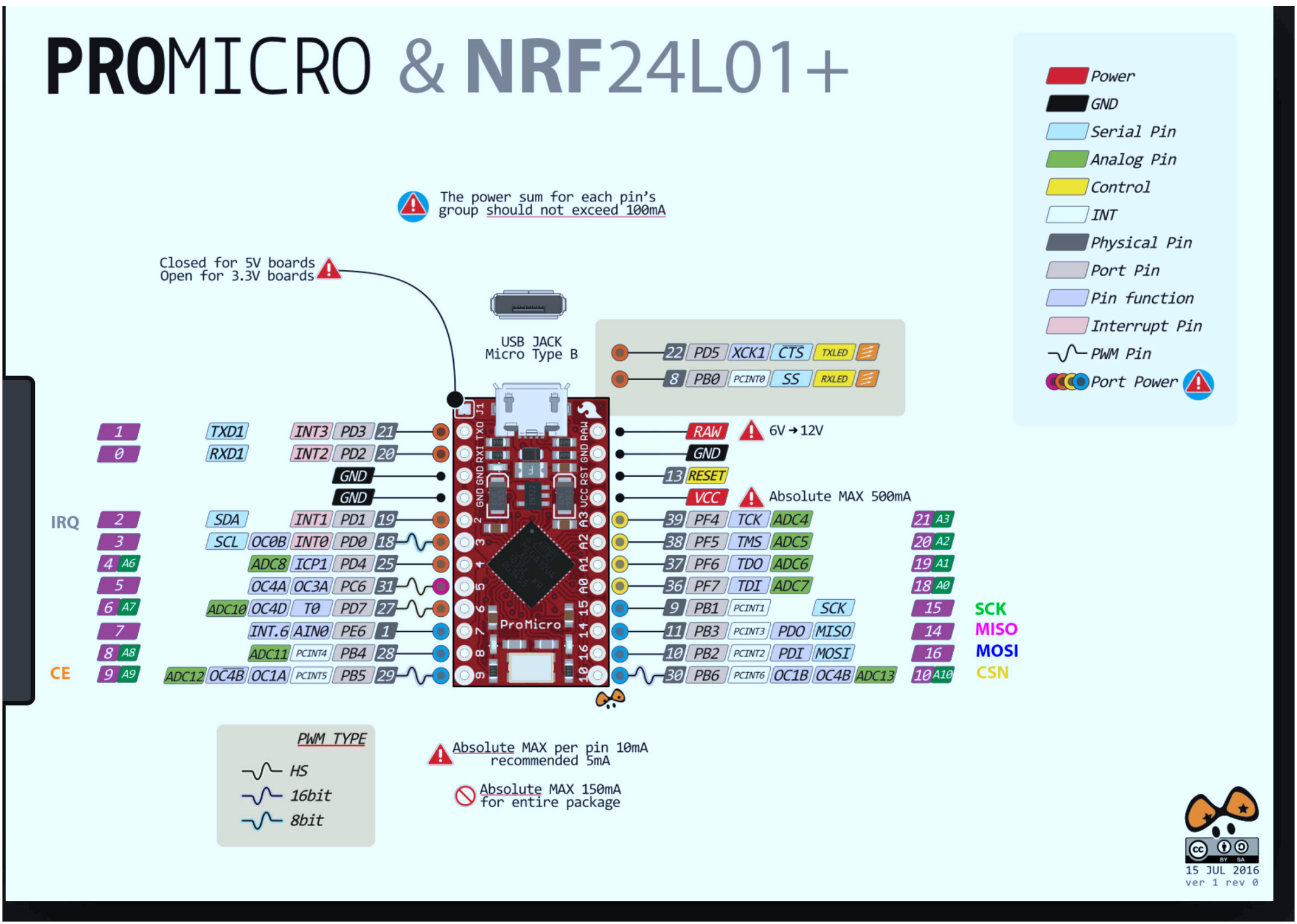
03 FEB 2013

PRO MINI PINOUT

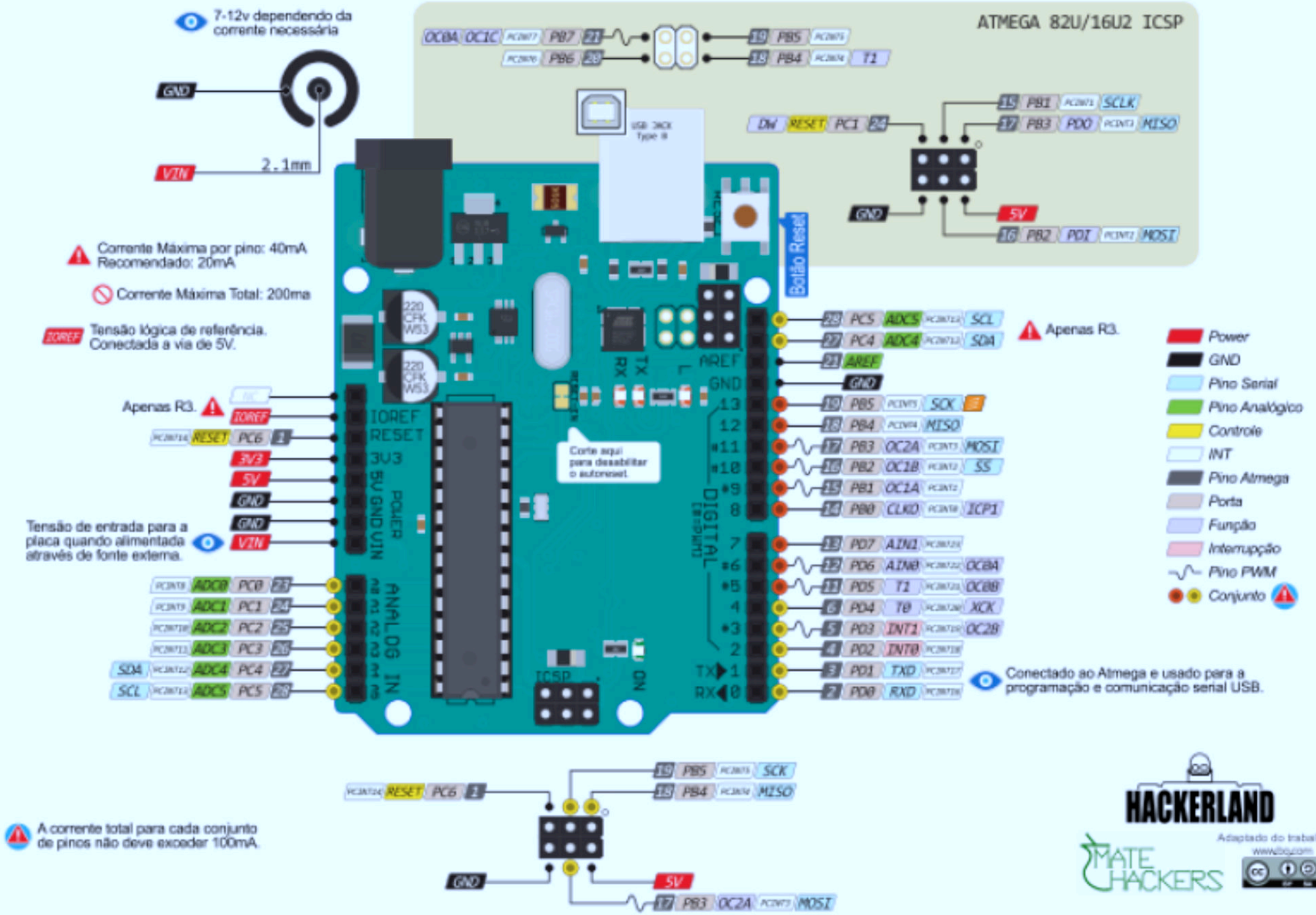


03 SEP 2014
ver 3 rev 1

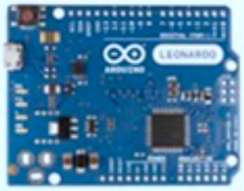
PROMICRO & NRF24L01+



UNO PINOUT


7-12v dependendo da corrente necessária
GND
VIN
2.1mm
ATMEGA 82U/16U2 ICSP
Botão Reset
Corte aqui para desabilitar o autoreset.
Corrente Máxima por pino: 40mA
Recomendado: 20mA
Corrente Máxima Total: 200ma
IOREF
Tensão lógica de referência. Conectada a via de 5V.
Apenas R3.
Tensão de entrada para a placa quando alimentada através de fonte externa.
POWER
ANALOG IN
DIGITAL (PWM~)
ICSP
Power
GND
Pino Serial
Pino Analógico
Controle
INT
Pino Atmega
Porta
Função
Interrupção
Pino PWM
Conjunto
Conectado ao Atmega e usado para a programação e comunicação serial USB.
A corrente total para cada conjunto de pinos não deve exceder 100mA.
HACKERLAND
Adaptado do trabalho de:
MATE HACKERS

Arduino Uno

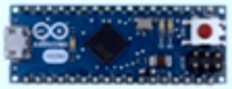


Arduino Leonardo

Arduino Due

Arduino Yún

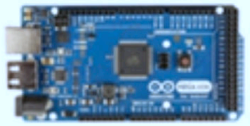
Arduino Tre



Arduino Micro

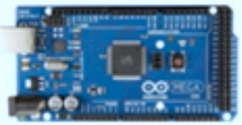
duino Robot

Arduino Esplora

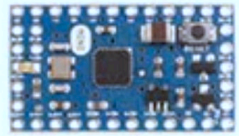


Arduino Mega ADK

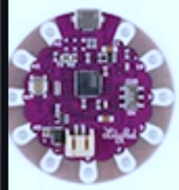
Arduino Ethernet



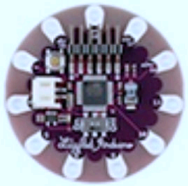
Arduino Mega 2560



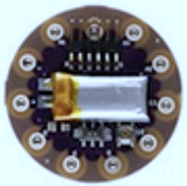
Arduino Mini



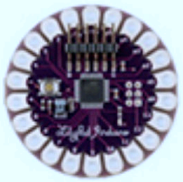
LilyPad Arduino USB



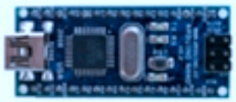
LilyPad Arduino Simple



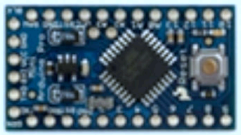
LilyPad Arduino SimpleSnap



LilyPad Arduino



Arduino Nano



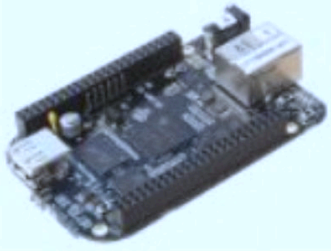
Arduino Pro Mini

Raspberry Pi 3
Model B (#1)

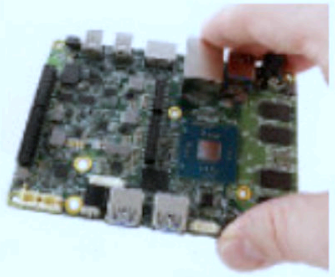
Raspberry Pi
Zero W (#2)

Raspberry Pi 2
Model B (#3)

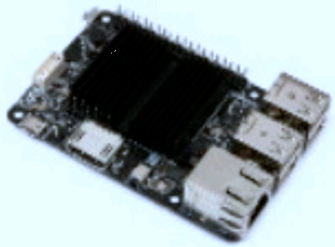
Odroid-XU4 (#4)



BeagleBone Black
Rev C (#5)



Udoo X86 (#6)



Odroid-C2 (#7)

DragonBoard
410c (#8)

Raspberry Pi Zero
(#9)

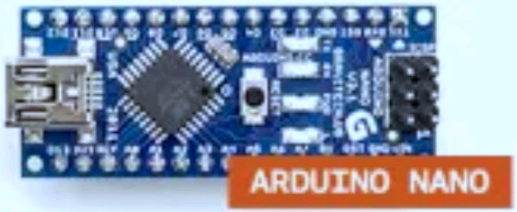
Arduino
Industrial 101
(#10)

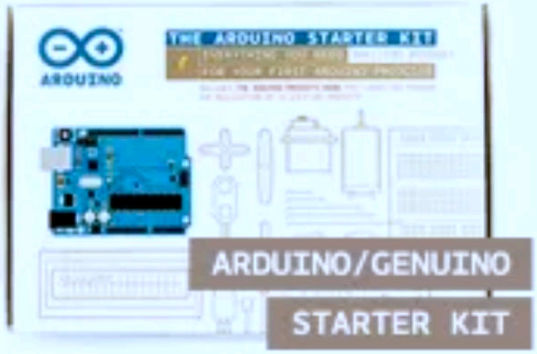

ARDUINO/GENUINO UNO


ARDUINO PRO

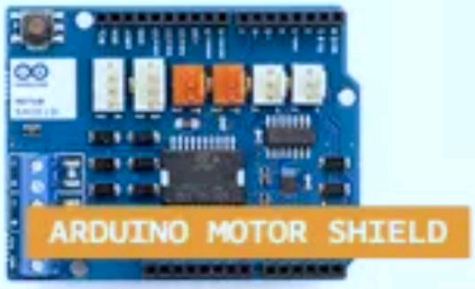

ARDUINO/GENUINO MICRO


ARDUINO PRO MINI

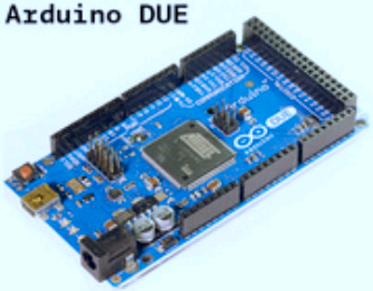

ARDUINO NANO


THE ARDUINO STARTER KIT
ARDUINO/GENUINO
STARTER KIT

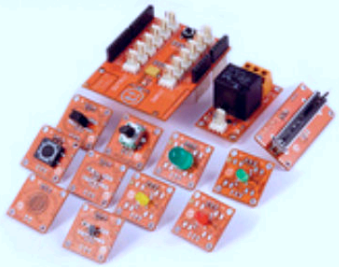

BASIC KIT
ARDUINO
ARDUINO BASIC KIT


ARDUINO MOTOR SHIELD

ARDUINO™

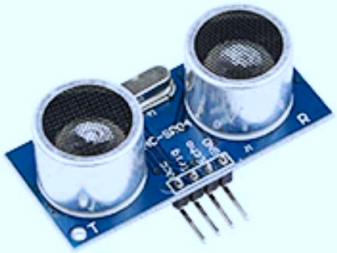
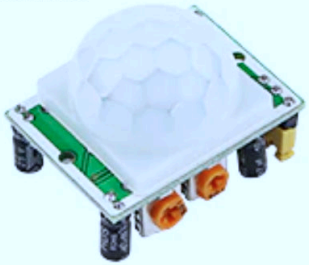
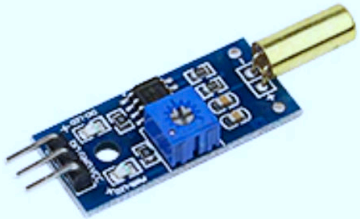
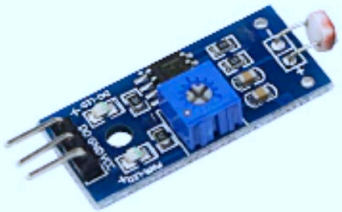
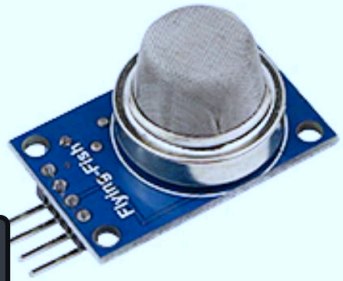
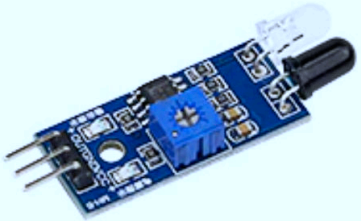
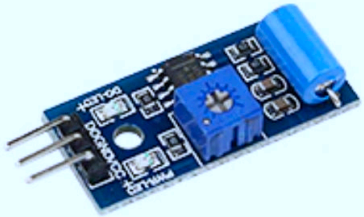
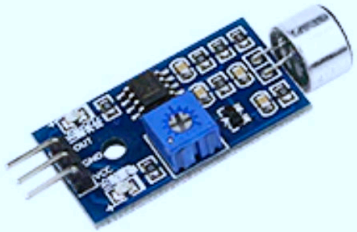
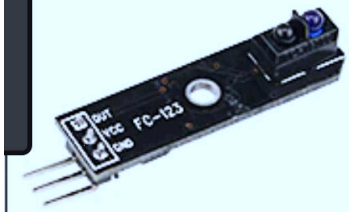
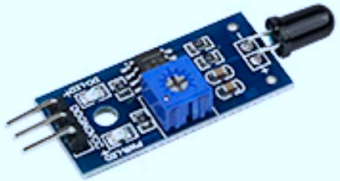
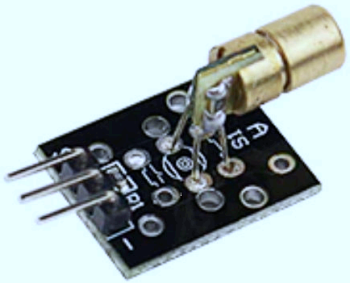
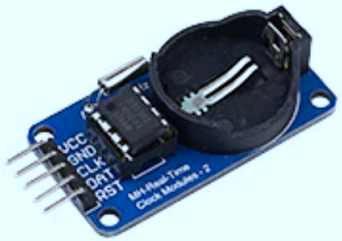
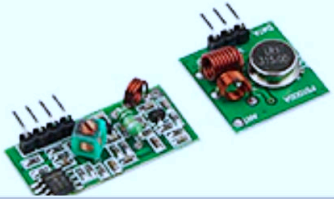
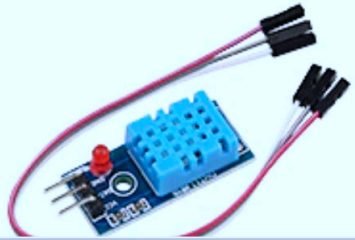
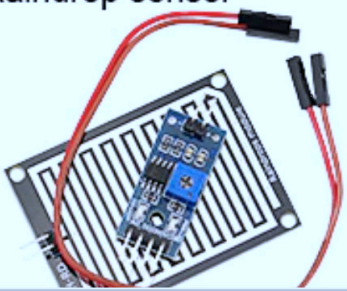
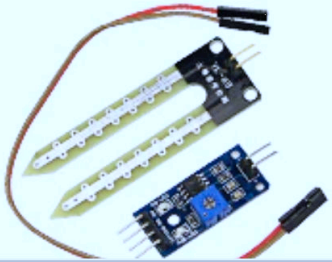
	ADK	Ethernet	Leonardo	Arduino DUE
				
Microcontroller	ATmega2560	ATmega328	ATmega32U4	Atmel SAM3U4E ARM Cortex M3
Clock	16 MHz	16 MHz	16 MHz	96 MHz
Flash Memory	256 KB	32 KB	32 KB	256 KB
SRAM	8 KB	2 KB	3.3 KB	50 KB
Digital I/O Pins	54	14 (10)	14	54
Analog Pins	16	6	6	16 (12bit)
	Android ADK Compatible USB Host **Develop your own android accessory!**	Wiznet W5100 Ethernet interface Optional PoE Module **Bring your project online!**	Onboard USB controller **Build your own USB devices!**	Onboard dual-channel DAC **Bringing 32 bit power to Arduino!**

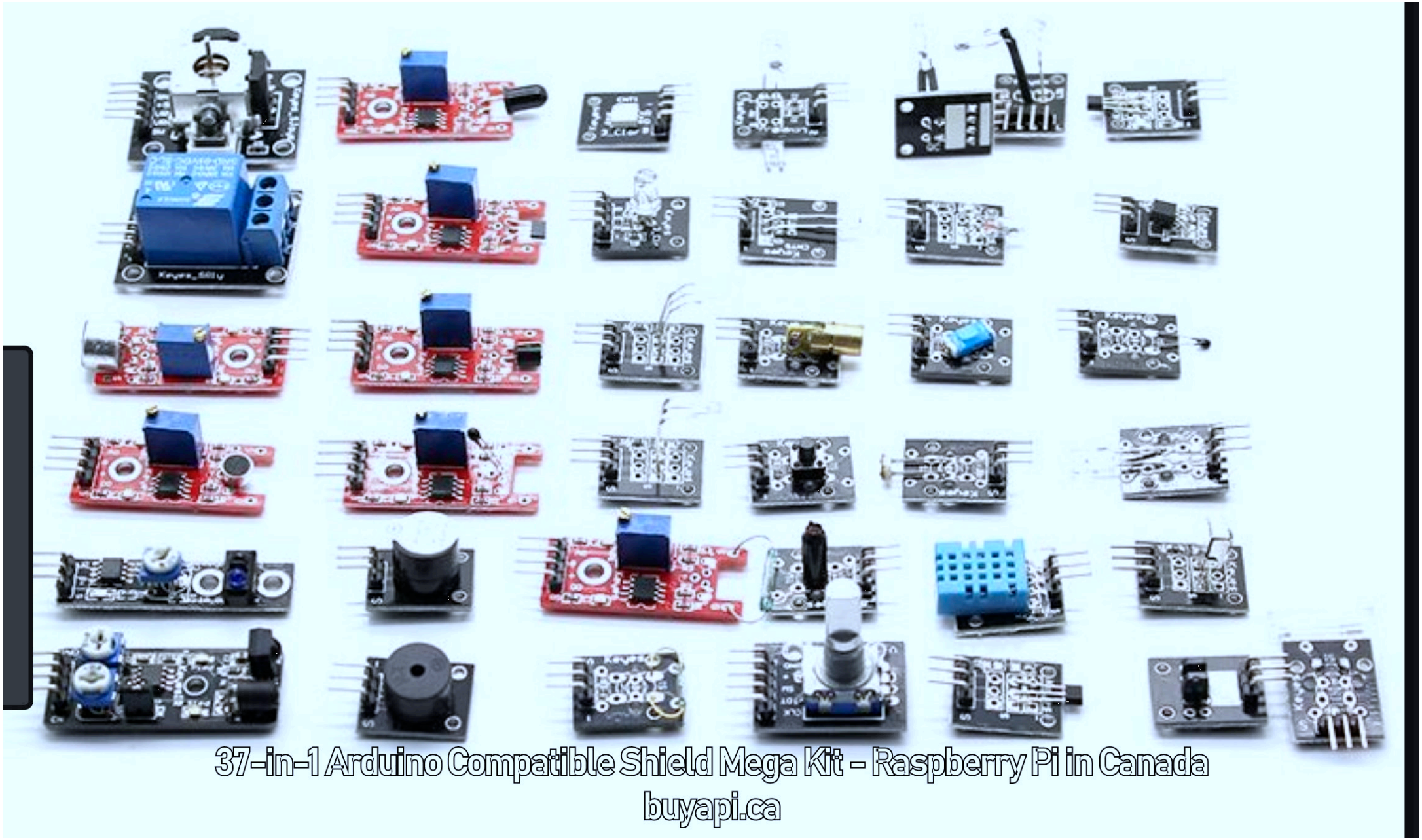
NEW PRODUCTS

	Wifi	TinkerKit	Arduino Robot System
			
	Avr32 co-processor with fully open-source firmware	Breadboard-free electronic prototyping	Arduino based dual platform robot
	H&D wifi module	30+ different modules	Multiprocessor
	Easy to upgrade firmware **Fully Hackable!**	**Easy to use instructions & tutorials!**	TinkerKit-compatible **Program your own behaviours!**

Look for us at Maker Faire 2011/New York

www.arduino.cc

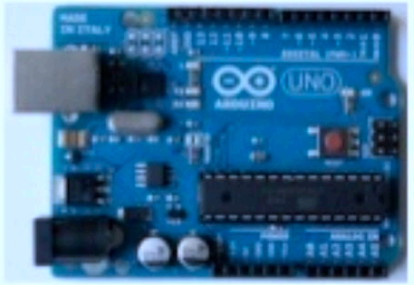
Ultrasonic module 	Human body sensor module 	Tilt sensor 	Photosensitive sensor 
Smoke sensor 	Infrared barrier sensor 	Vibration sensor 	Sound sensor 
path search sensor 	Flame sensor 	Laser Head Sensor 	Clock module 
Super regenerative module 	Temperature and humidity sensor 	Raindrop sensor 	Soil Sensors 

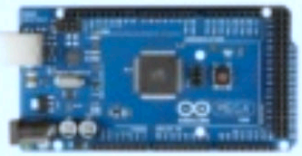


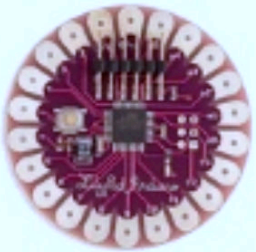
37-in-1 Arduino Compatible Shield Mega Kit – Raspberry Pi in Canada
buyapi.ca

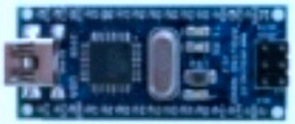
Types of Arduino

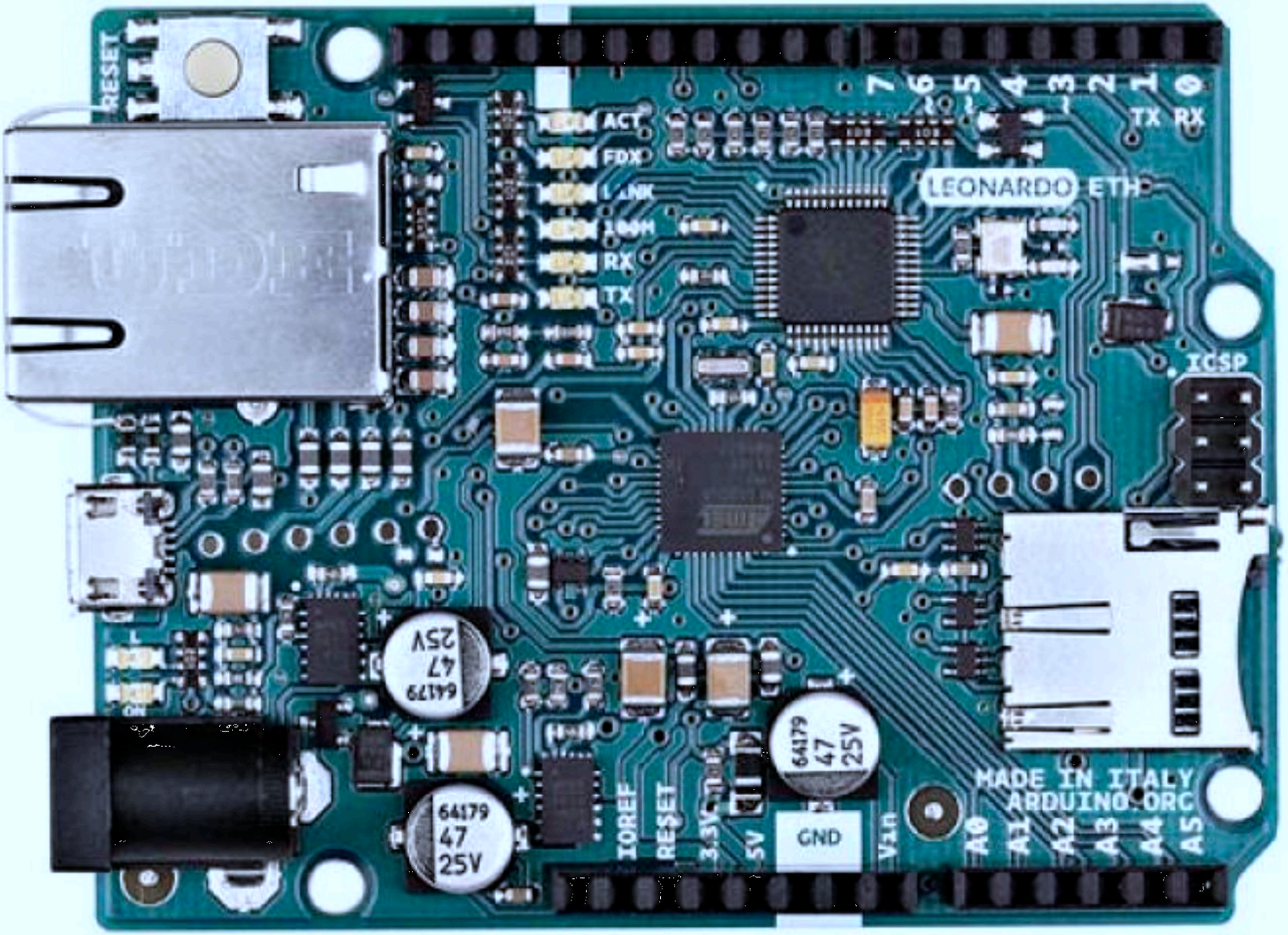
- Uno - most popular - all rounder
- Mega - more pins
- LilyPad - wearable
- Leonardo
- Mini
- Nano
- Bluetooth
- Pro Versions

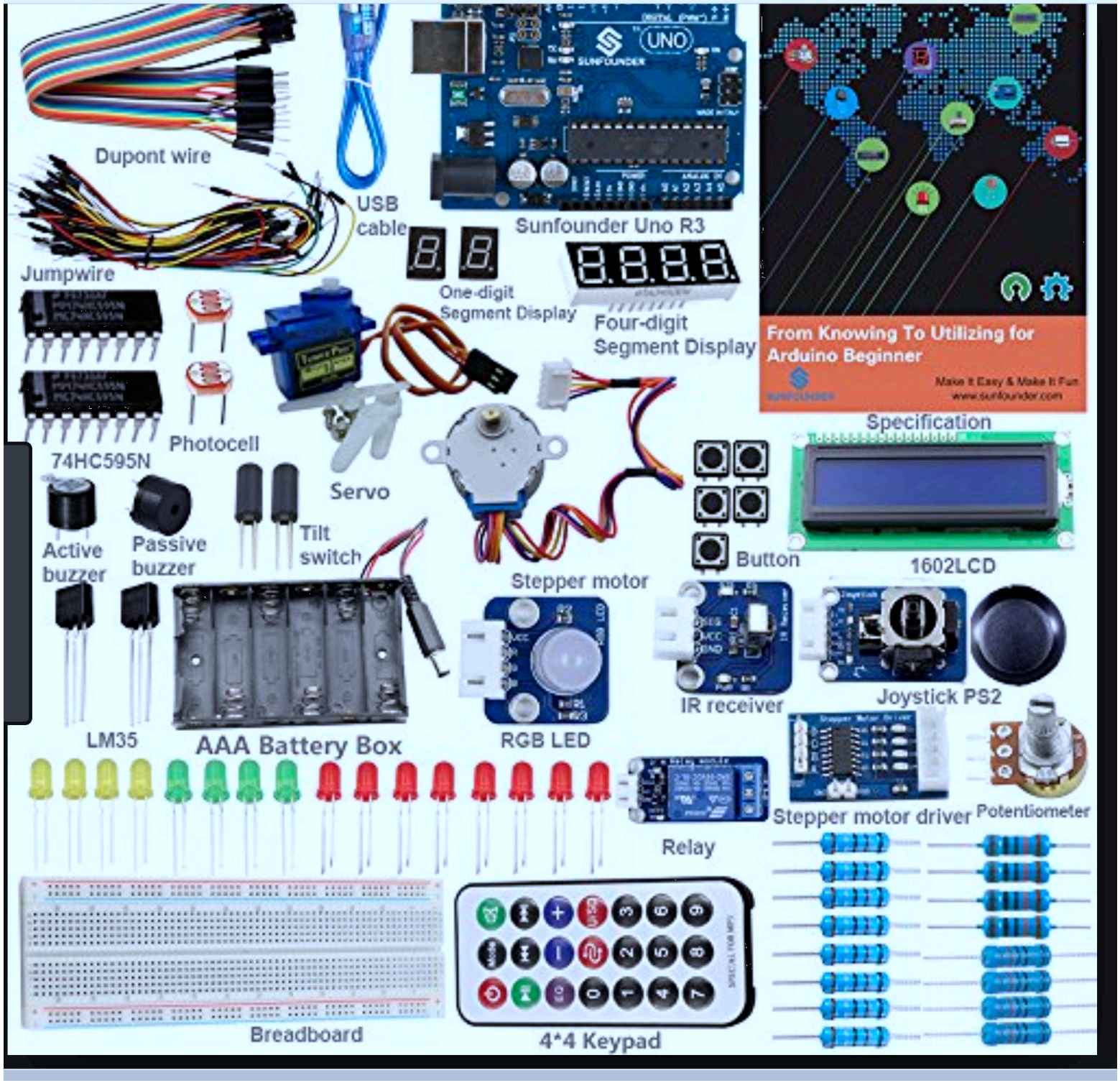


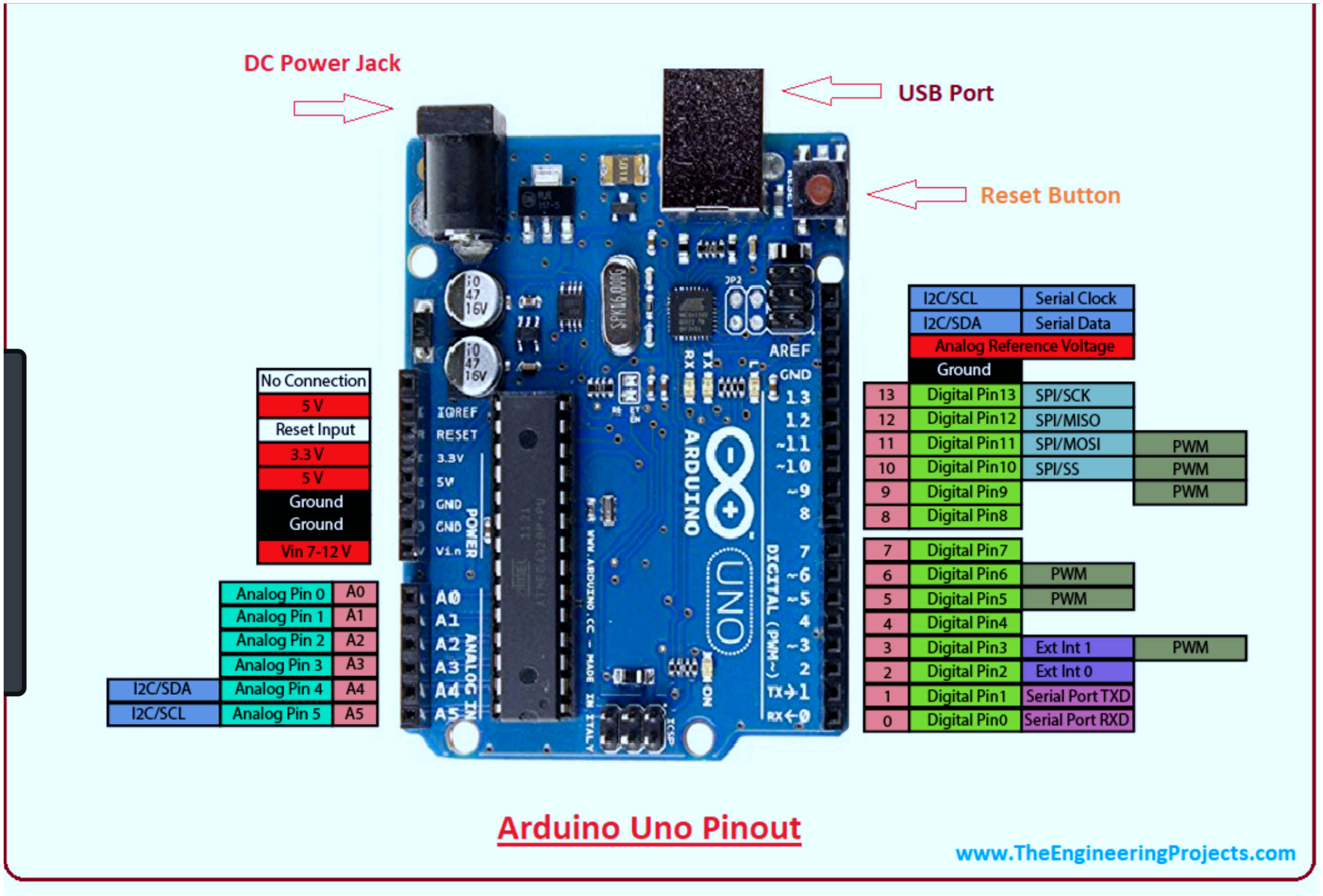




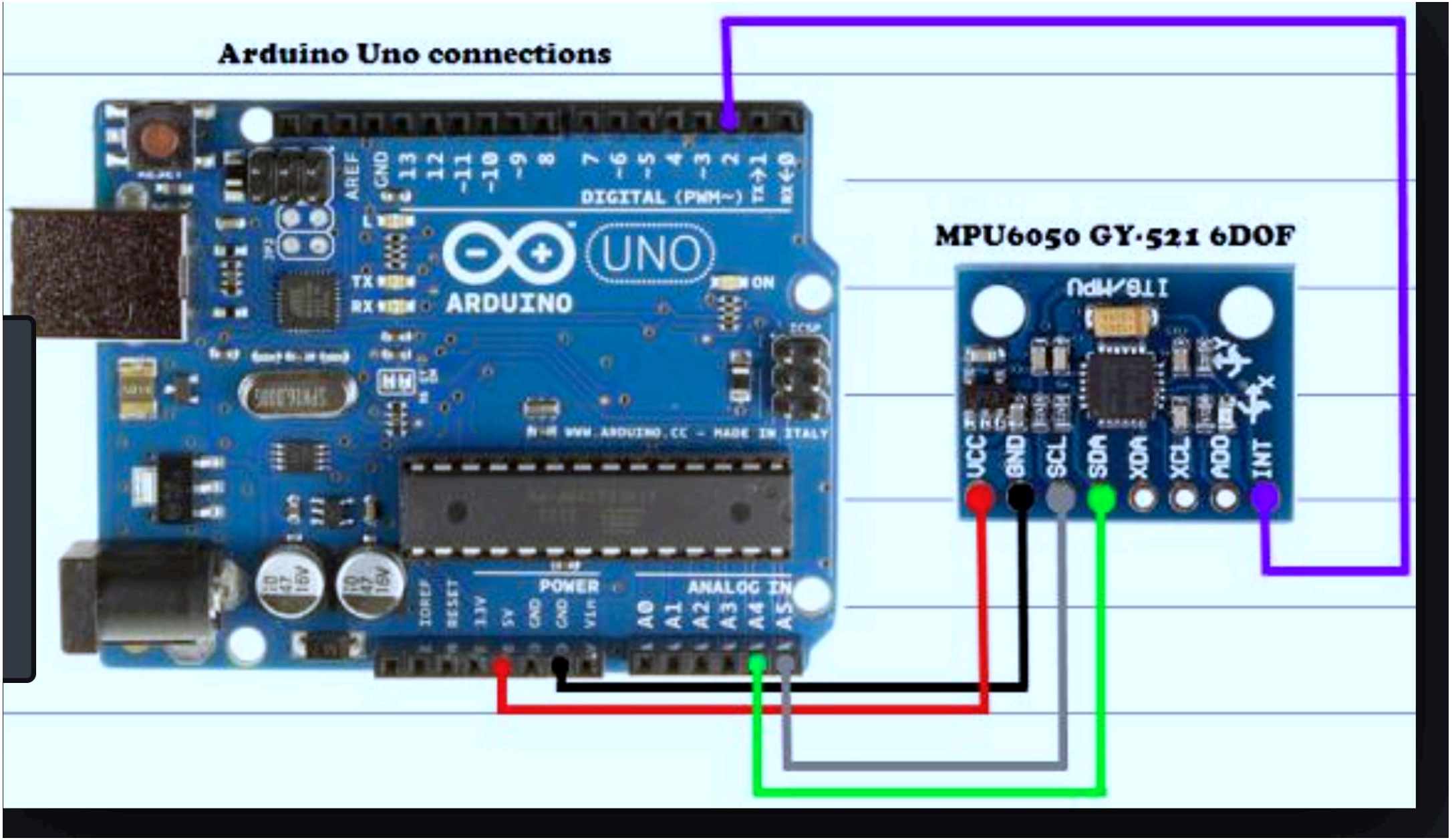


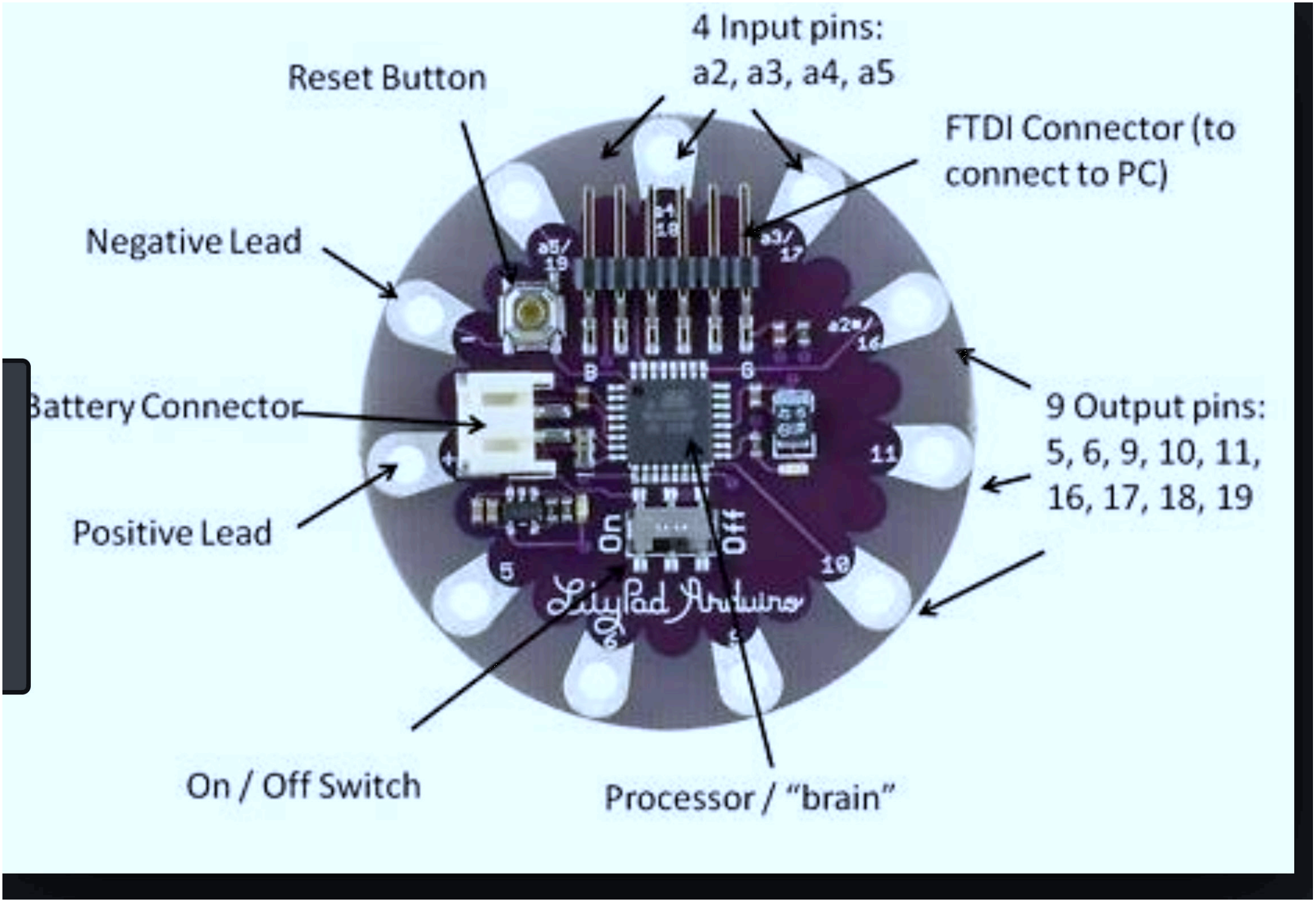

RESET
ACT
FDX
LINK
100M
RX
TX
7 6 5 4 3 2 1 0
TX RX
LEONARDO ETH
ICSP
64179
47
25V
IOREF
RESET
3.3V
5V
GND
Vin
MADE IN ITALY
ARDUINO.ORC
A0
A1
A2
A3
A4
A5

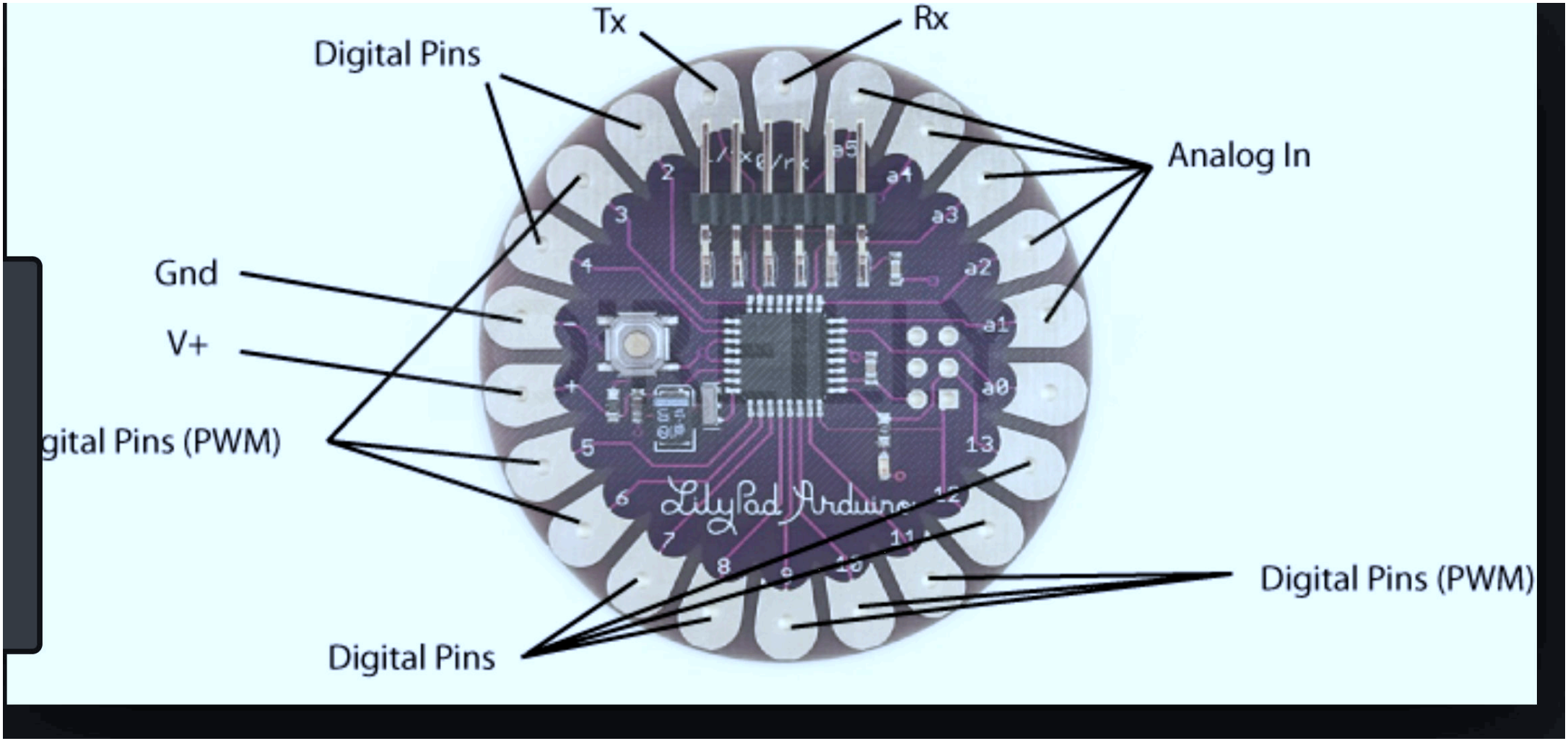

Dupont wire
USB cable
Sunfounder Uno R3
Jumpwire
One-digit Segment Display
Four-digit Segment Display
From Knowing To Utilizing for Arduino Beginner
Make It Easy & Make It Fun
www.sunfounder.com
Specification
74HC595N
Photocell
Servo
Active buzzer
Passive buzzer
Tilt switch
Stepper motor
Button
1602LCD
IR receiver
Joystick PS2
LM35
AAA Battery Box
RGB LED
Stepper motor driver
Potentiometer
Relay
Breadboard
4*4 Keypad

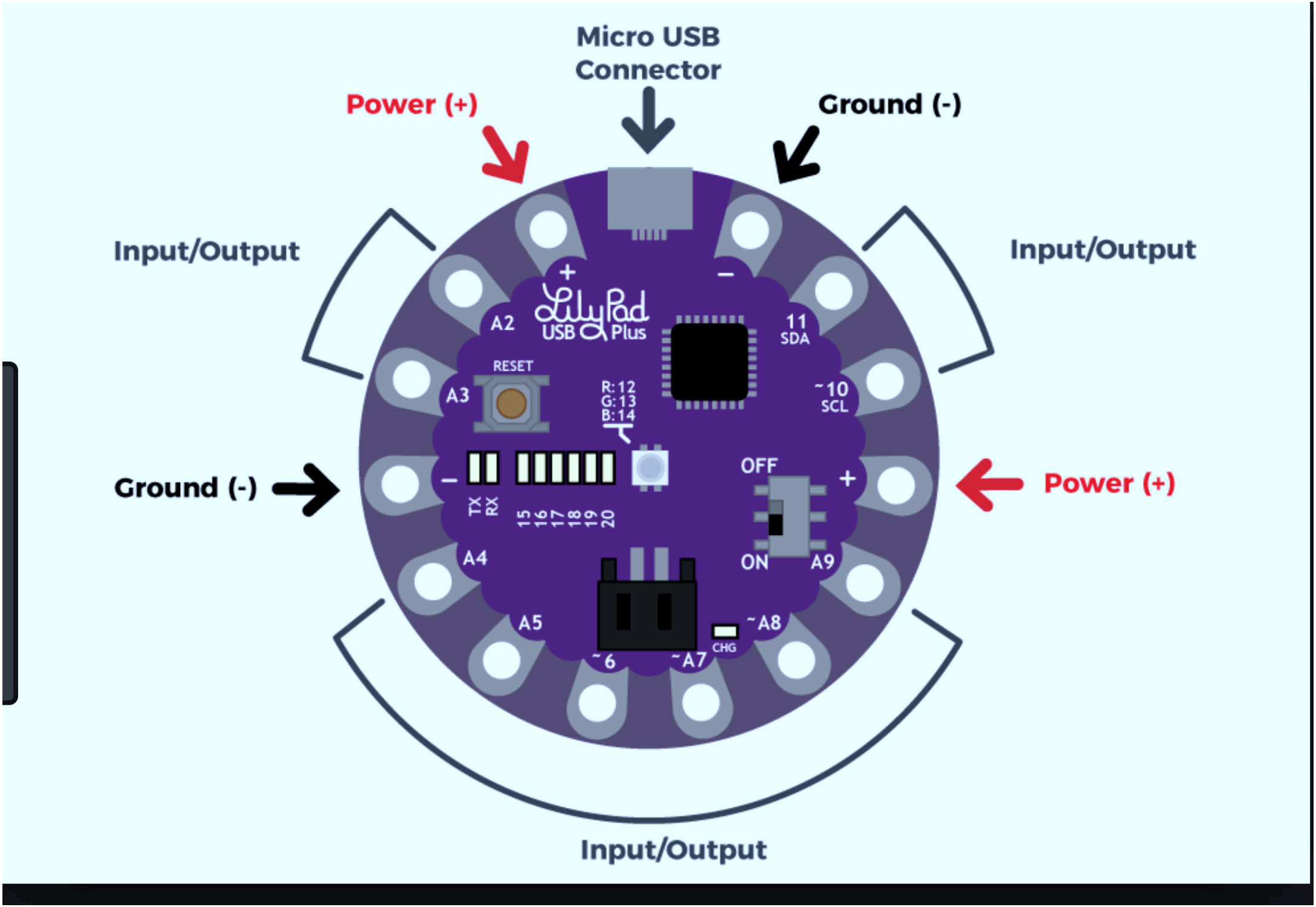

DC Power Jack
USB Port
Reset Button
No Connection
5 V
Reset Input
3.3 V
5 V
Ground
Ground
Vin 7-12 V
Analog Pin 0 A0
Analog Pin 1 A1
Analog Pin 2 A2
Analog Pin 3 A3
I2C/SDA Analog Pin 4 A4
I2C/SCL Analog Pin 5 A5
I2C/SCL Serial Clock
I2C/SDA Serial Data
Analog Reference Voltage
Ground
13 Digital Pin13 SPI/SCK
12 Digital Pin12 SPI/MISO
11 Digital Pin11 SPI/MOSI PWM
10 Digital Pin10 SPI/SS PWM
9 Digital Pin9 PWM
8 Digital Pin8
7 Digital Pin7
6 Digital Pin6 PWM
5 Digital Pin5 PWM
4 Digital Pin4
3 Digital Pin3 Ext Int 1 PWM
2 Digital Pin2 Ext Int 0
1 Digital Pin1 Serial Port TXD
0 Digital Pin0 Serial Port RXD
Arduino Uno Pinout
www.TheEngineeringProjects.com

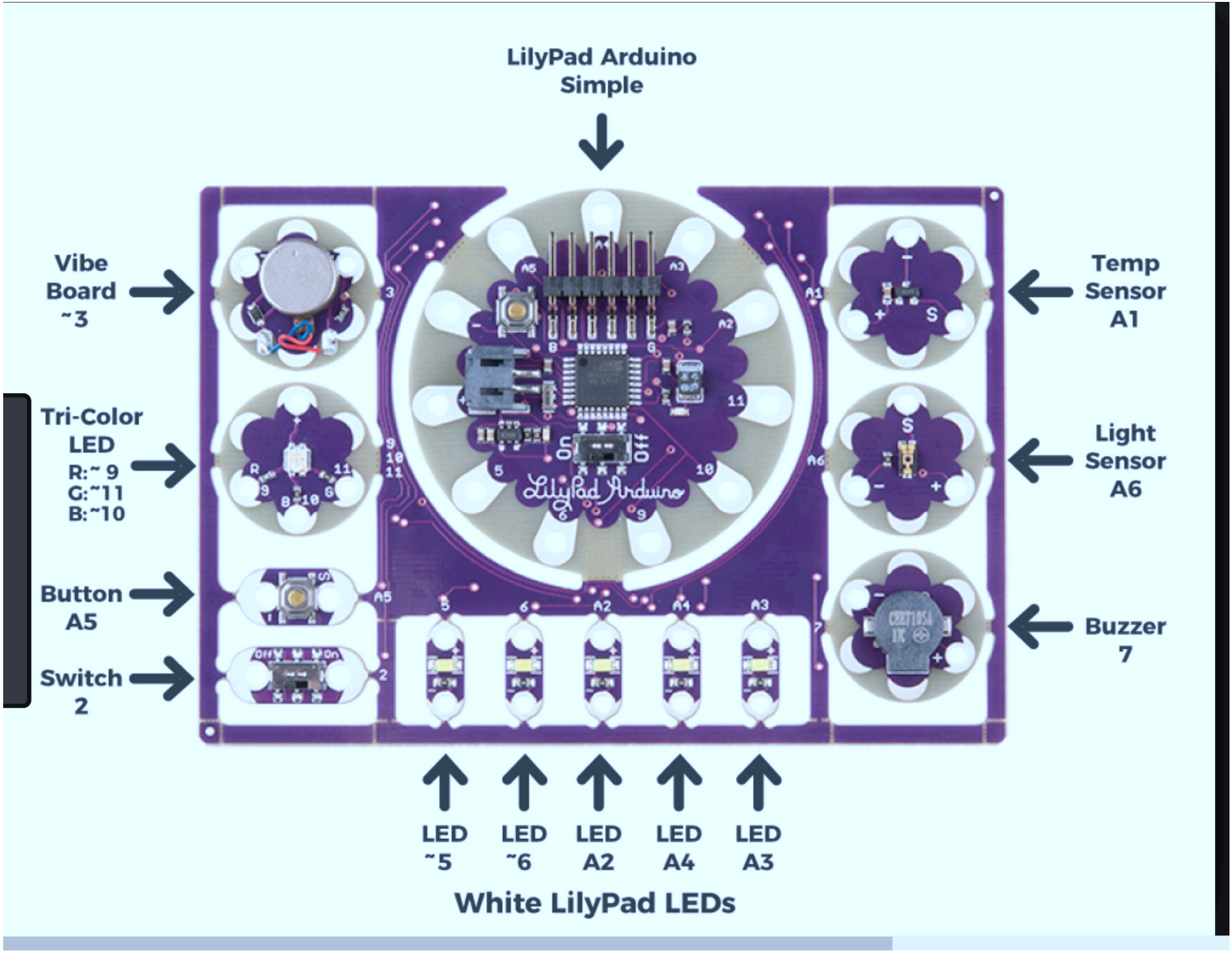
ARDUINO MEGA

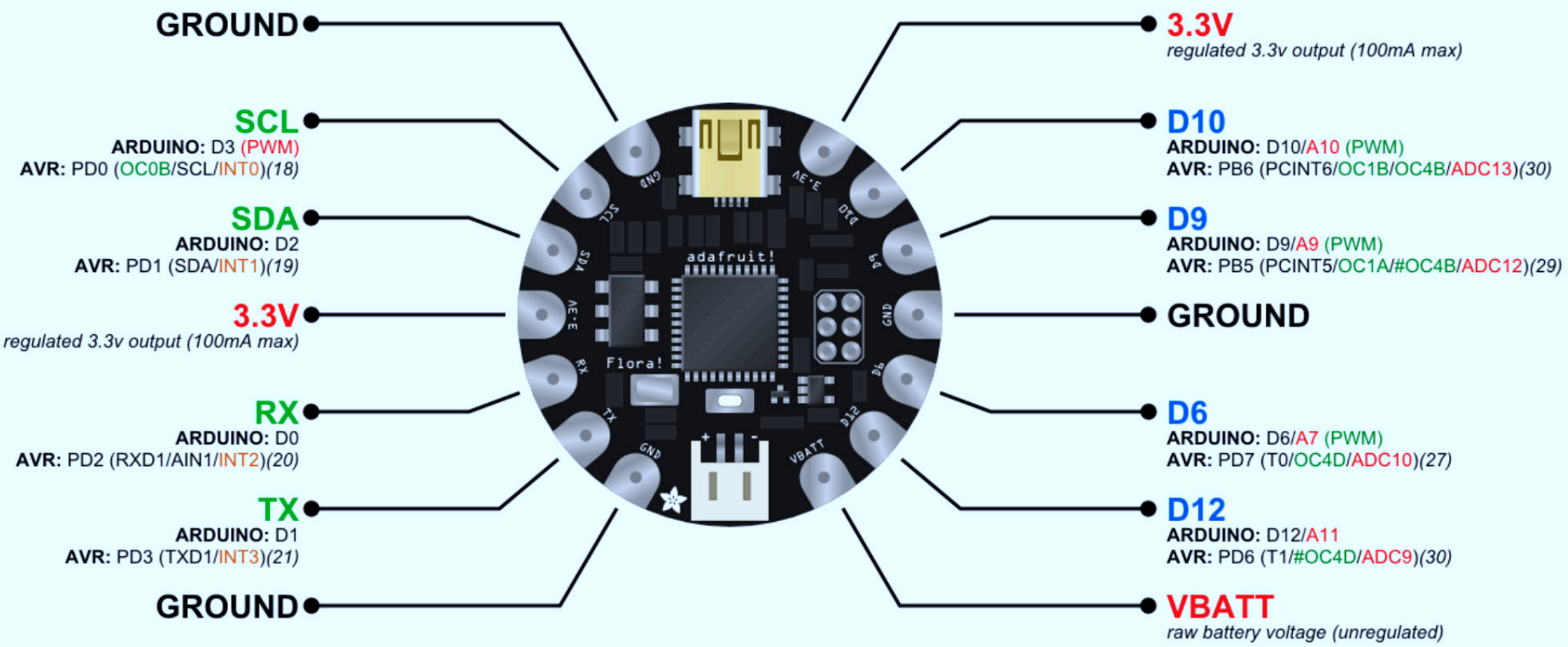

Arduino Uno connections
MPU6050 GY-521 6DOF
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
VCC
GND
SCL
SDA
XDA
XCL
ADO
INT


4 Input pins:
a2, a3, a4, a5
Reset Button
FTDI Connector (to connect to PC)
Negative Lead
Battery Connector
9 Output pins:
5, 6, 9, 10, 11, 16, 17, 18, 19
Positive Lead
On / Off Switch
Processor / "brain"
LilyPad Arduino


Tx
Rx
Digital Pins
Analog In
Gnd
V+
gital Pins (PWM)
Digital Pins (PWM)
Digital Pins
LilyPad Arduino


Micro USB
Connector
Power (+)
Ground (-)
Input/Output
Input/Output
Ground (-)
Power (+)
Input/Output
LilyPad
USB Plus
A2
11
SDA
RESET
A3
R:12
G:13
B:14
~10
SCL
OFF
TX
RX
15
16
17
18
19
20
ON
A9
A4
A5
~A8
CHG
~6
~A7


LilyPad Arduino
Simple
Vibe
Board
~3
Tri-Color
LED
R:~ 9
G:~11
B:~10
Button
A5
Switch
2
Temp
Sensor
A1
Light
Sensor
A6
Buzzer
7
LED
~5
LED
~6
LED
A2
LED
A4
LED
A3
White LilyPad LEDs



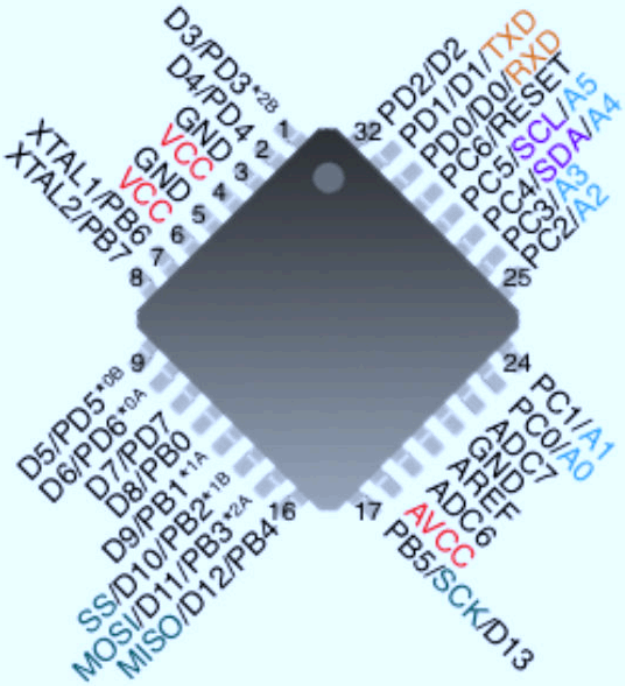
FLORA Wearable Electronics Platform

adafruit.com/products/659

赤 akafugu.jp

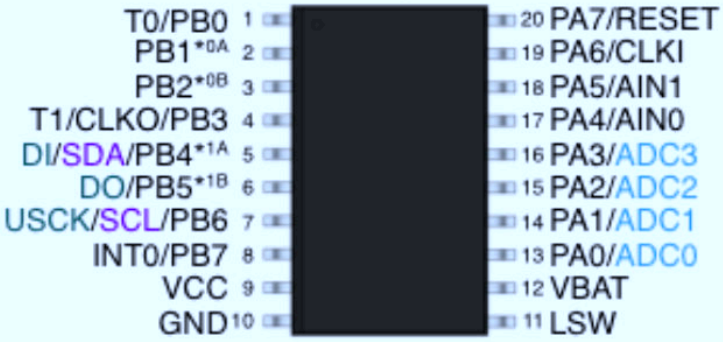
Microcontroller Reference Sheet v1.0 SMD edition

ATMega8/48/88/168/328/Arduino



ATTiny4/5/9/10

ATTiny43u

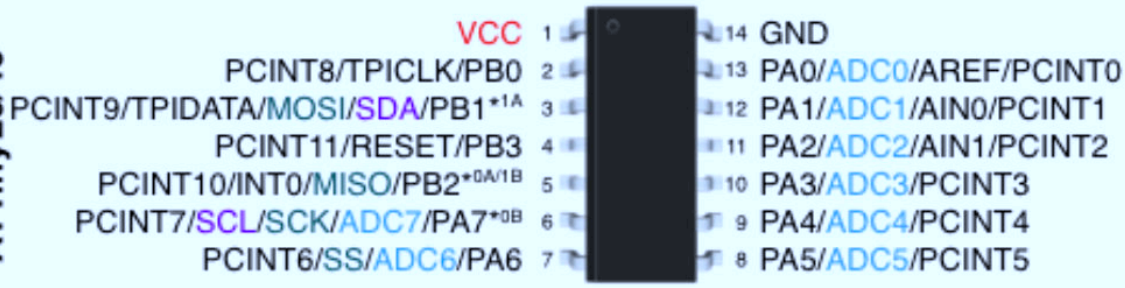


ATTiny87/167

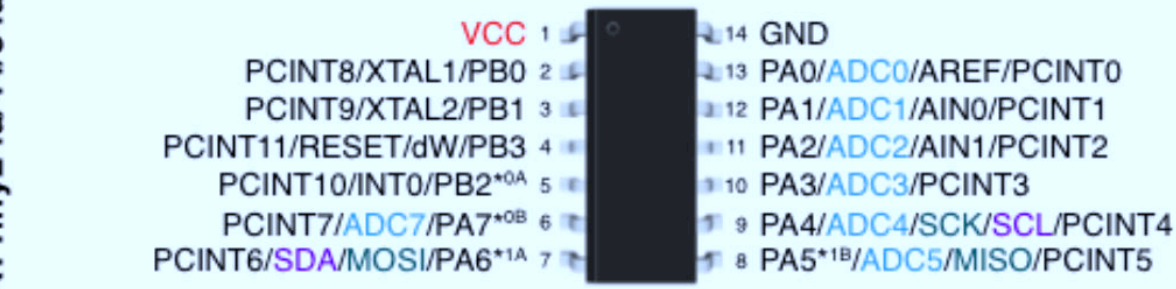
ISP Header

NOTES:
- PWM pins are marked with (*)

ATTiny20/40



ATTiny24a/44/84a



JavaScript

Methods

Object
toString
toLocaleString
valueOf
hasOwnProperty
isPrototypeOf
propertyIsEnumerable

String
charAt
charCodeAt
fromCharCode
concat
indexOf
lastIndexOf
localeCompare
match
replace
search
slice
split
substring
substr
toLowerCase
toUpperCase
toLocaleLowerCase
toLocaleUpperCase

RegEx
test
match
exec

Array
concat
join
push
pop
reverse
shift
slice
sort
splice
unshift

Number
toFixed
toExponential
toPrecision

Date
parse
toDateString
toTimeString
getDate
getDay
getFullYear
getHours
getMilliseconds
getMinutes
getMonth
getSeconds
getTime
getTimezoneOffset
getYear
setDate
setHours
setMilliseconds
setMinutes
setMonth
setSeconds
setYear
toLocaleTimeString

XMLHttpRequest

Safari, Mozilla, Opera:

```
var req = new XMLHttpRequest();
```

Internet Explorer:

```
var req = new
ActiveXObject("Microsoft.XMLHTTP");
```

XMLHttpRequest Object Methods

abort()
getAllResponseHeaders()
getResponseHeader(header)
open(method, URL)
send(body)
setRequestHeader(header, value)

XMLHttpRequest Object Properties

onreadystatechange
readyState
responseText
responseXML
status
statusText

XMLHttpRequest readyState Values

0	Uninitiated
1	Loading
2	Loaded
3	Interactive
4	Complete

JAVASCRIPT IN HTML

External JavaScript File

```
<script type="text/javascript"
src="javascript.js"></script>
```

Inline JavaScript

```
<script type="text/javascript">
  <!--
    // JavaScript Here
  //-->
</script>
```

Functions

Window
alert
blur
clearTimeout
close
focus
open
print
setTimeout

Built In
eval
parseInt
parseFloat
isNaN
isFinite
decodeURI
decodeURIComponent
encodeURI
encodeURIComponent
escape
unescape

REGULAR EXPRESSIONS - FORMAT

Regular expressions in JavaScript take the form:

var RegEx = **/pattern/modifiers**;

REGULAR EXPRESSIONS - MODIFIERS

/g	Global matching
/i	Case insensitive
/s	Single line mode
/m	Multi line mode

REGULAR EXPRESSIONS - PATTERNS

^	Start of string
$	End of string
.	Any single character
(a|b)	a or b
(...)	Group section
[abc]	Item in range (a or b or c)
[^abc]	Not in range (not a or b or c)
a?	Zero or one of a
a*	Zero or more of a
a+	One or more of a
a{3}	Exactly 3 of a
a{3,}	3 or more of a
a{3,6}	Between 3 and 6 of a
!(pattern)	"Not" prefix. Apply rule when URL does not match pattern.

EVENT HANDLERS

onAbort	onMouseDown
onBlur	onMouseMove
onChange	onMouseOut
onClick	onMouseOver
onDblClick	onMouseUp
onDragDrop	onMove
onError	onReset
onFocus	onResize
onKeyDown	onSelect
onKeyPress	onSubmit
onKeyUp	onUnload
onLoad	

FUNCTIONS AND METHODS

A method is a type of function, associated with an object. A normal function is not associated with an object.

DOM Methods

Document
clear
createDocument
createDocumentFragment
createElement
createEvent
createEventObject
createRange
createTextNode
getElementsByTagName
getElementById
write

Node
addEventListener
appendChild
attachEvent
cloneNode
createTextRange
detachEvent
dispatchEvent
fireEvent
getAttributeNS
getAttributeNode
hasChildNodes
hasAttribute
hasAttributes
insertBefore
removeChild
removeEventListener
replaceChild
scrollIntoView

Form
submit

DOM Collections
item

Range
collapse
createContextualFragment
moveEnd
moveStart
parentElement
select
setStartBefore

Style
getPropertyValue
setProperty

Event
initEvent
preventDefault
stopPropagation

XMLSerializer
serializeToString

XMLHTTP
open
send

XMLDOM
loadXML

DOMParser
parseFromString

IMPERIAL & METRIC CONVERSION

Millimeters x 0.03937 = inches
Centimeters x 0.3937 = inches
Meters x 3.281 = feet
Kilometers x 0.6214 = miles

METRIC POWERS OF A THOUSAND

Amount	Multiplier	Prefix/Abbr
Trillion	1,000,000,000,000	tera- (T)
Billion	1,000,000,000	giga- (G)
Million	1,000,000	mega- (M)
Thousand	1,000	kilo- (k)
Thousandth	0.001	milli- (m)
Millionth	0.000001	micro- (µ/u)
Billionth	0.000000001	nano (n)
Trillionth	0.000000000001	pico (p)

MISSING UNITS

Type	Units
Battery	volts (V)
Large capacitor	microfarads (µF)
Small capacitor	picofarads (pF)
Crystal	megahertz (MHz)
Inductor	henrys (H)
Oscillator	megahertz (MHz)
Potentiometer	ohms (Ω)
Resistor	ohms (Ω)

UNIT ABBREVIATIONS

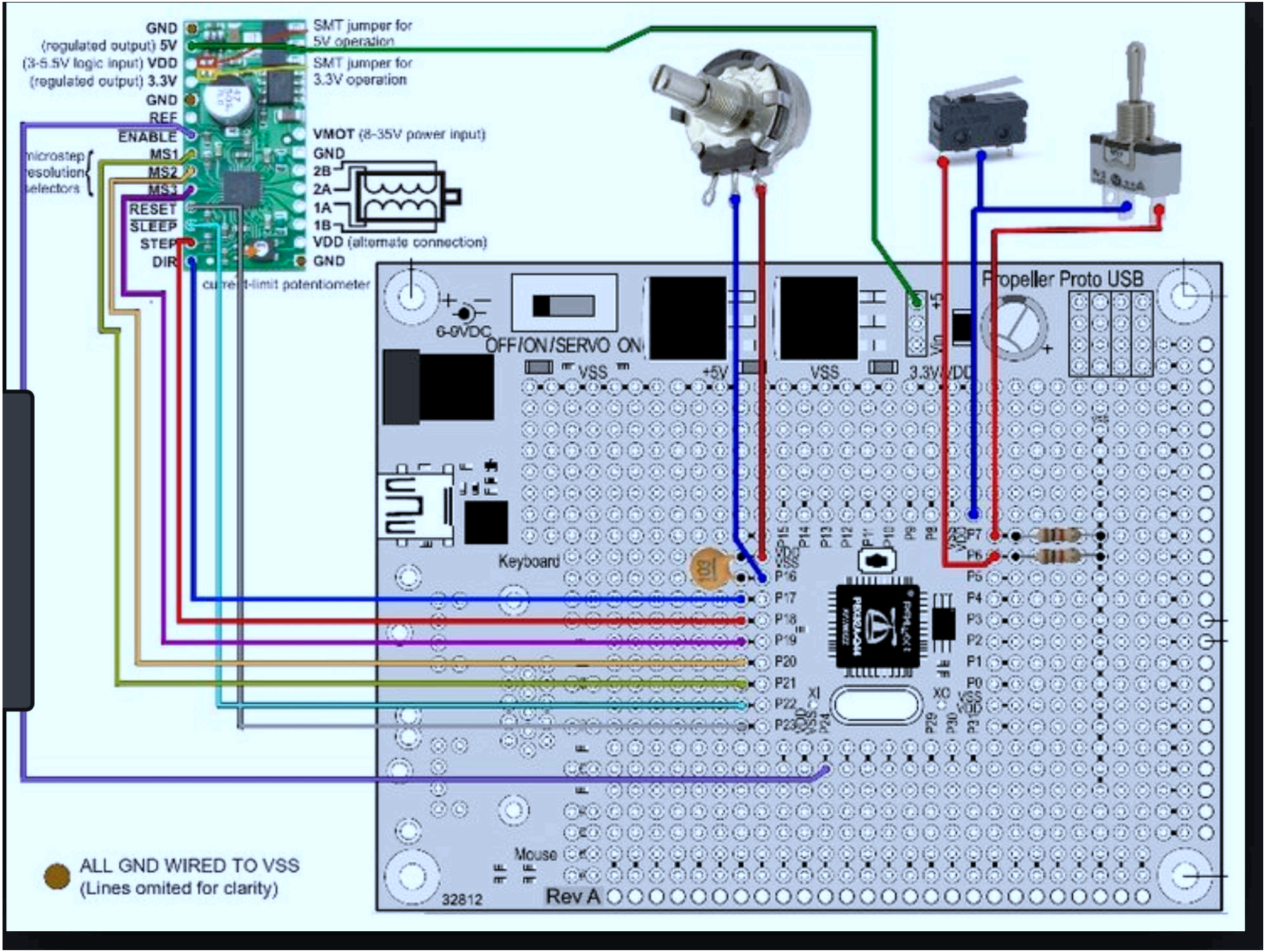
Measurement	Units	Abbr
Length	meter	m
Resistance	ohm	Ω
Voltage	volt	V
Current	ampere or amp	A
Power	watt	W
Frequency	hertz	Hz
Capacitance	farad	F
Inductance	henry	G
Amplification	beta factor	h_{fe} or β
Luminous Intensity	candela	cd
Mass	gram	g
Torque	newton-meter	Nm
Capacity	amp-hour	Ah
Temperature	Celsius	°C
Rotational speed	revolution per min	rpm

COLOR BAND VALUES

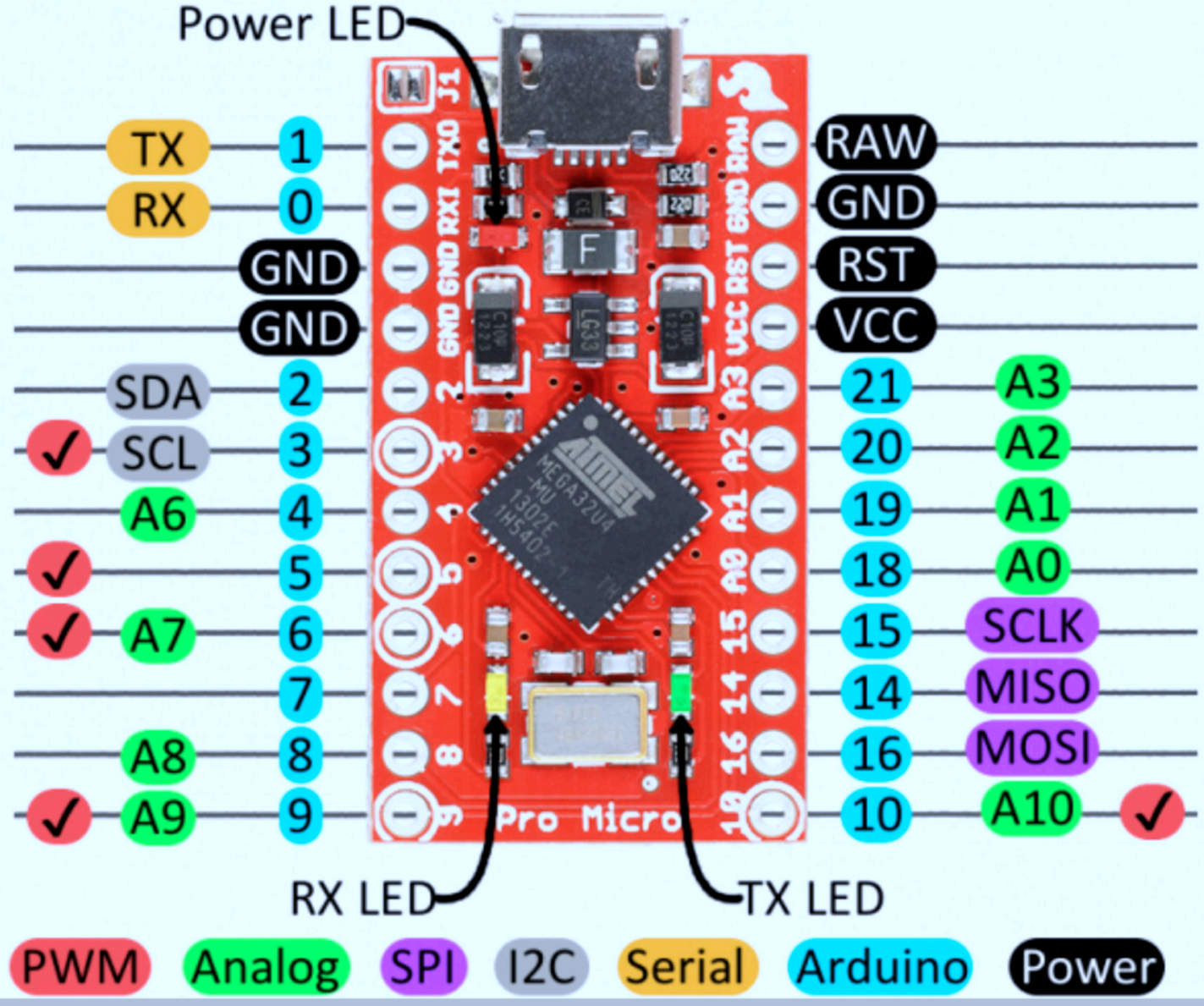
Color	1st Band	2nd Band	3rd Band
Black	0	0	x1
Brown	1	1	x10
Red	2	2	x100
Orange	3	3	x1,000
Yellow	4	4	x10,000
Green	5	5	x100,000
Blue	6	6	x1,000,000
Violet	7	7	...
Gray	8	8	...
White	9	9	...

COLOR TOLERANCE BANDS

Color	Tolerance
(none)	20%
Silver	10%
Gold	5%
Red	2%
Brown	1%


GND
(regulated output) 5V
(3-5.5V logic input) VDD
(regulated output) 3.3V
GND
REF
ENABLE
MS1
MS2
MS3
RESET
SLEEP
STEP
DIR
microstep resolution selectors
SMT jumper for 5V operation
SMT jumper for 3.3V operation
VMOT (8-35V power input)
GND
2B
2A
1A
1B
VDD (alternate connection)
GND
current-limit potentiometer
Propeller Proto USB
6-9VDC
OFF/ON/SERVO ON
VSS
+5V
VSS
3.3V/VDD
Vin
+5
Keyboard
Mouse
P16
P17
P18
P19
P20
P21
P22
P23
P7
P6
P5
P4
P3
P2
P1
P0
XI
XO
32812
Rev A
ALL GND WIRED TO VSS
(Lines omited for clarity)

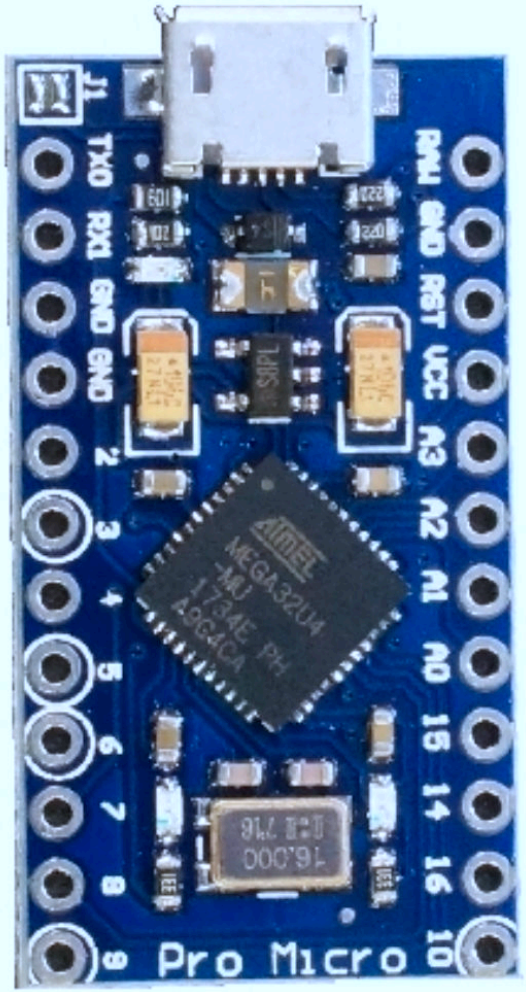
All of the Pro Micro's I/O and power pins are broken out to two, parallel headers. Some pins are for power input or output, other pins are dedicated I/O pins. Further, the I/O pins can have special abilities, like analog input. Here's a map of which pin is where, and what special hardware functions it may have:



Proto Supplies

Pro Micro Pinout

USB Micro-B Port
To Computer



Serial Port TXD / Ext Int 3 / Digital Pin 1 **(1)**

Serial Port RXD / Ext Int 2 / Digital Pin 0 **(0)**

Ground

Ground

(I2C) SDA / Ext Int 1 / Digital Pin 2 **(2)**

(I2C) SCL / Ext Int 0 / PWM / Digital Pin 3 **(3)**

Analog Pin 6 / Digital Pin 4 **(4)**

PWM / Digital Pin 5 **(5)**

PWM / Analog 7 / Digital Pin 6 **(6)**

Ext Int 6 / Digital Pin 7 **(7)**

Analog Pin 8 / Digital Pin 8 **(8)**

PWM / Analog Pin 9 / Digital Pin 9 / PWM **(9)**

7-12V Power Input

Ground

Reset Input (Ground to reset board)

5V Output or Input

(A3) Analog Pin 3 / Digital Pin 21

(A2) Analog Pin 2 / Digital Pin 20

(A1) Analog Pin 1 / Digital Pin 19

(A0) Analog Pin 0 / Digital Pin 18

(15) Digital Pin 15 / (SPI) SCK

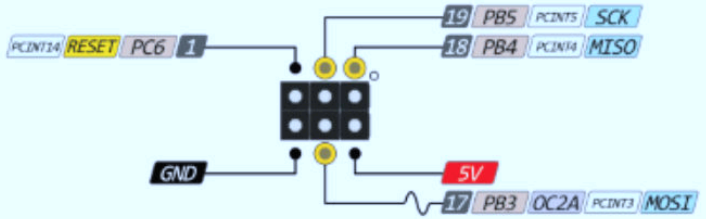
(14) Digital Pin 14 / PWM / (SPI) MISO

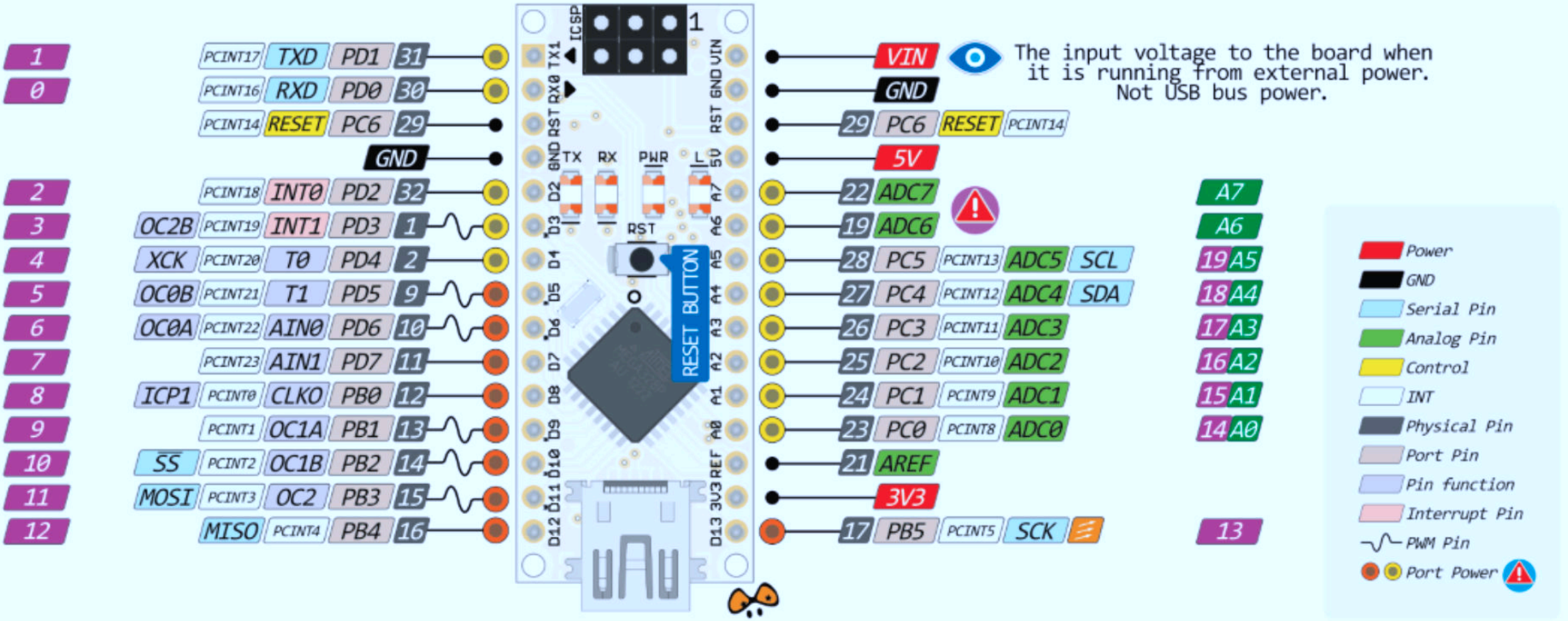
(16) Digital Pin 16 / (SPI) MOSI

(10) Digital Pin 10 / Analog Pin 10 / PWM

Red numbers in paranthesis are the name to use when referencing that pin.
Analog pins are references as A0 thru A3 even when using as digital I/O

NANO PINOUT



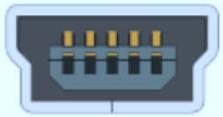


The input voltage to the board when it is running from external power. Not USB bus power.

- Power
- GND
- Serial Pin
- Analog Pin
- Control
- INT
- Physical Pin
- Port Pin
- Pin function
- Interrupt Pin
- PWM Pin
- Port Power

The power sum for each pin's group should not exceed 100mA

USB JACK
Mini Type B



Absolute MAX per pin 40mA
recommended 20mA

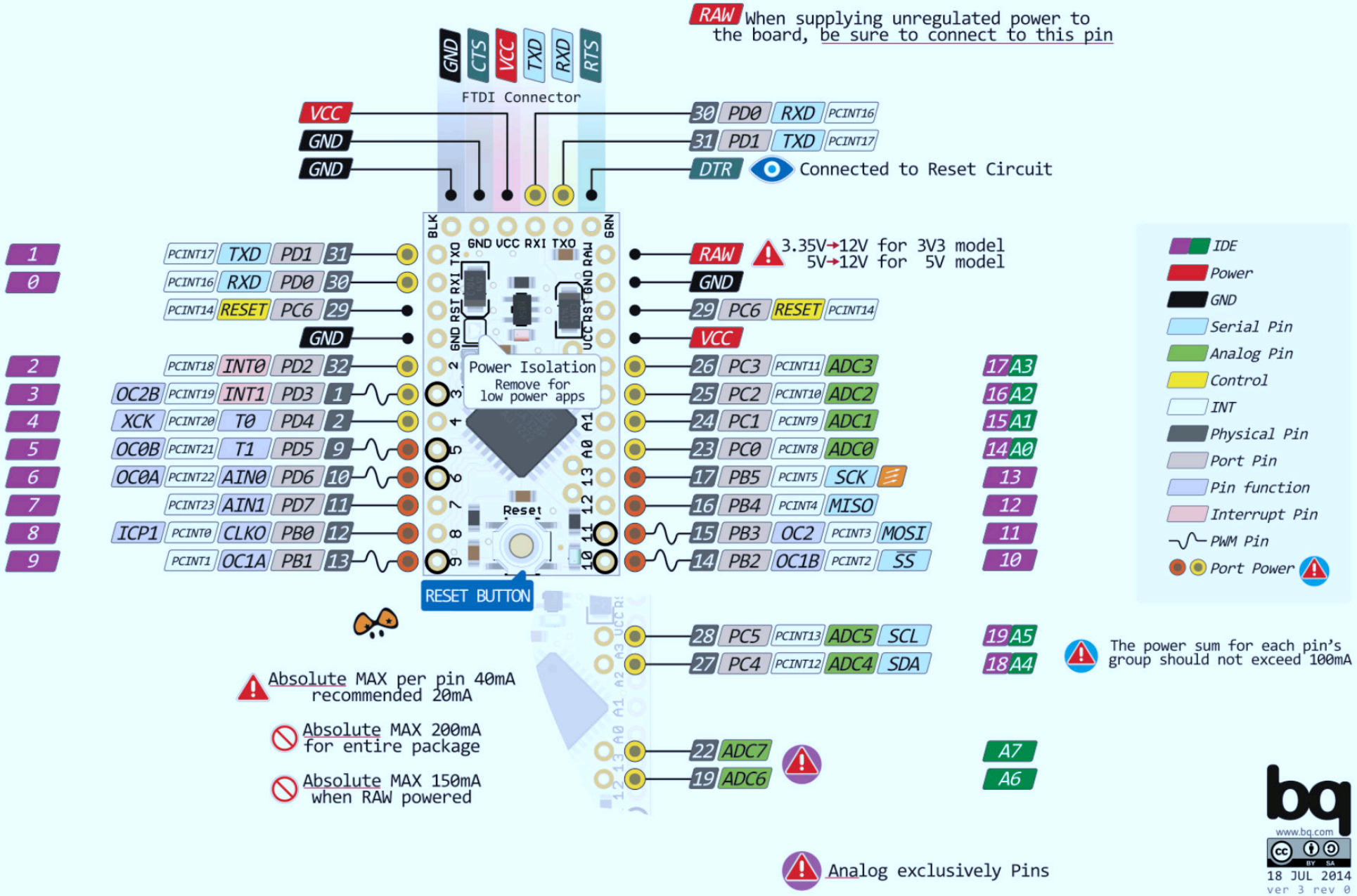
Absolute MAX 200mA
for entire package

Analog exclusively Pins

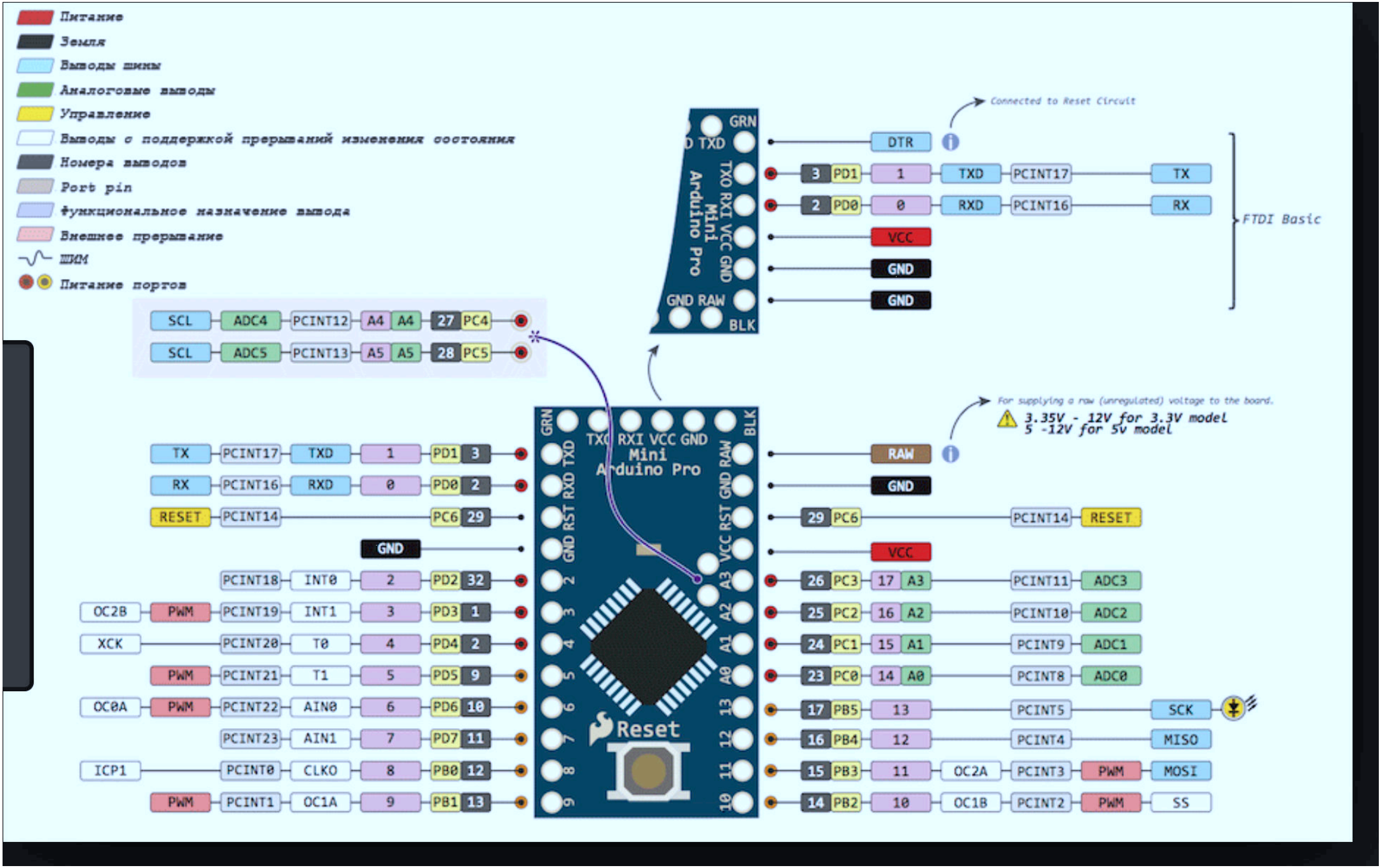
bq
www.bq.com

19 AUG 2014
ver 3 rev 1

PRO MINI PINOUT

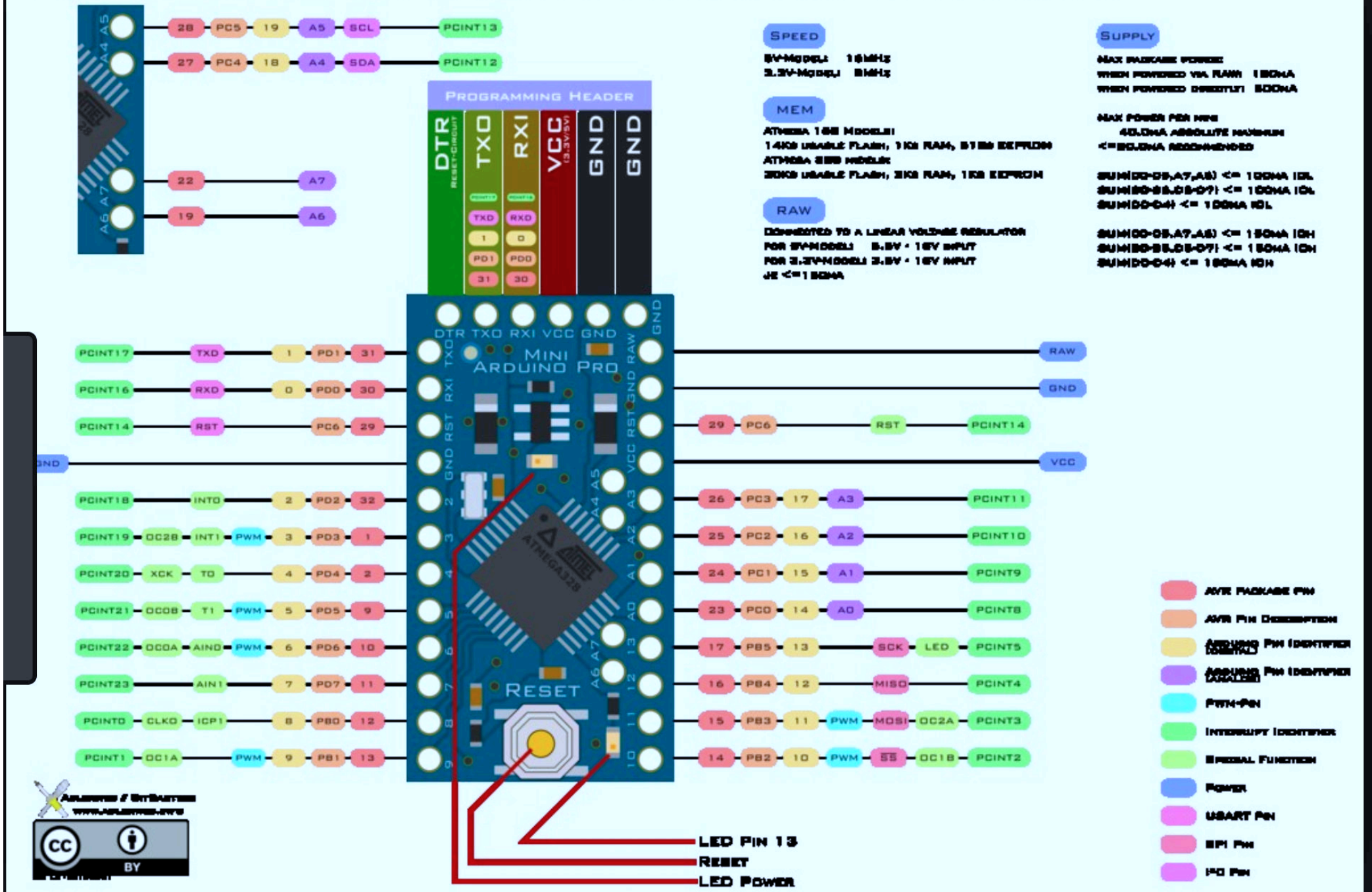


bq
www.bq.com
18 JUL 2014
ver 3 rev 0


Питание
Земля
Выводы шины
Аналоговые выводы
Управление
Выводы с поддержкой прерываний изменения состояния
Номера выводов
Port pin
Функциональное назначение вывода
Внешнее прерывание
ШИМ
Питание портов
Connected to Reset Circuit
DTR
3 PD1 1 TXD PCINT17 TX
2 PD0 0 RXD PCINT16 RX
VCC
GND
GND
FTDI Basic
GRN TXO RXI VCC GND BLK
Mini
Arduino Pro
GND RAW
SCL ADC4 PCINT12 A4 A4 27 PC4
SCL ADC5 PCINT13 A5 A5 28 PC5
For supplying a raw (unregulated) voltage to the board.
3.35V - 12V for 3.3V model
5 -12V for 5v model
RAW
GND
29 PC6 PCINT14 RESET
VCC
26 PC3 17 A3 PCINT11 ADC3
25 PC2 16 A2 PCINT10 ADC2
24 PC1 15 A1 PCINT9 ADC1
23 PC0 14 A0 PCINT8 ADC0
17 PB5 13 PCINT5 SCK
16 PB4 12 PCINT4 MISO
15 PB3 11 OC2A PCINT3 PWM MOSI
14 PB2 10 OC1B PCINT2 PWM SS
TX PCINT17 TXD 1 PD1 3
RX PCINT16 RXD 0 PD0 2
RESET PCINT14 PC6 29
GND
PCINT18 INT0 2 PD2 32
OC2B PWM PCINT19 INT1 3 PD3 1
XCK PCINT20 T0 4 PD4 2
PWM PCINT21 T1 5 PD5 9
OC0A PWM PCINT22 AIN0 6 PD6 10
PCINT23 AIN1 7 PD7 11
ICP1 PCINT0 CLKO 8 PB0 12
PWM PCINT1 OC1A 9 PB1 13
Reset

Arduino® Pro Mini Pinout Cheat Sheet

Warning! There are several models named 'Pro Mini' using different pinouts - check your device!



Speed

5V-Model: 16MHz
3.3V-Model: 8MHz

Mem

ATmega 168 Models:
14KB usable Flash, 1Kb RAM, 512b EEPROM
ATmega 328 models:
30KB usable Flash, 2Kb RAM, 1Kb EEPROM

Raw

Connected to a linear voltage regulator
for 5V-model: 5.5V - 16V input
for 3.3V-model: 3.6V - 16V input
je <=150mA

Supply

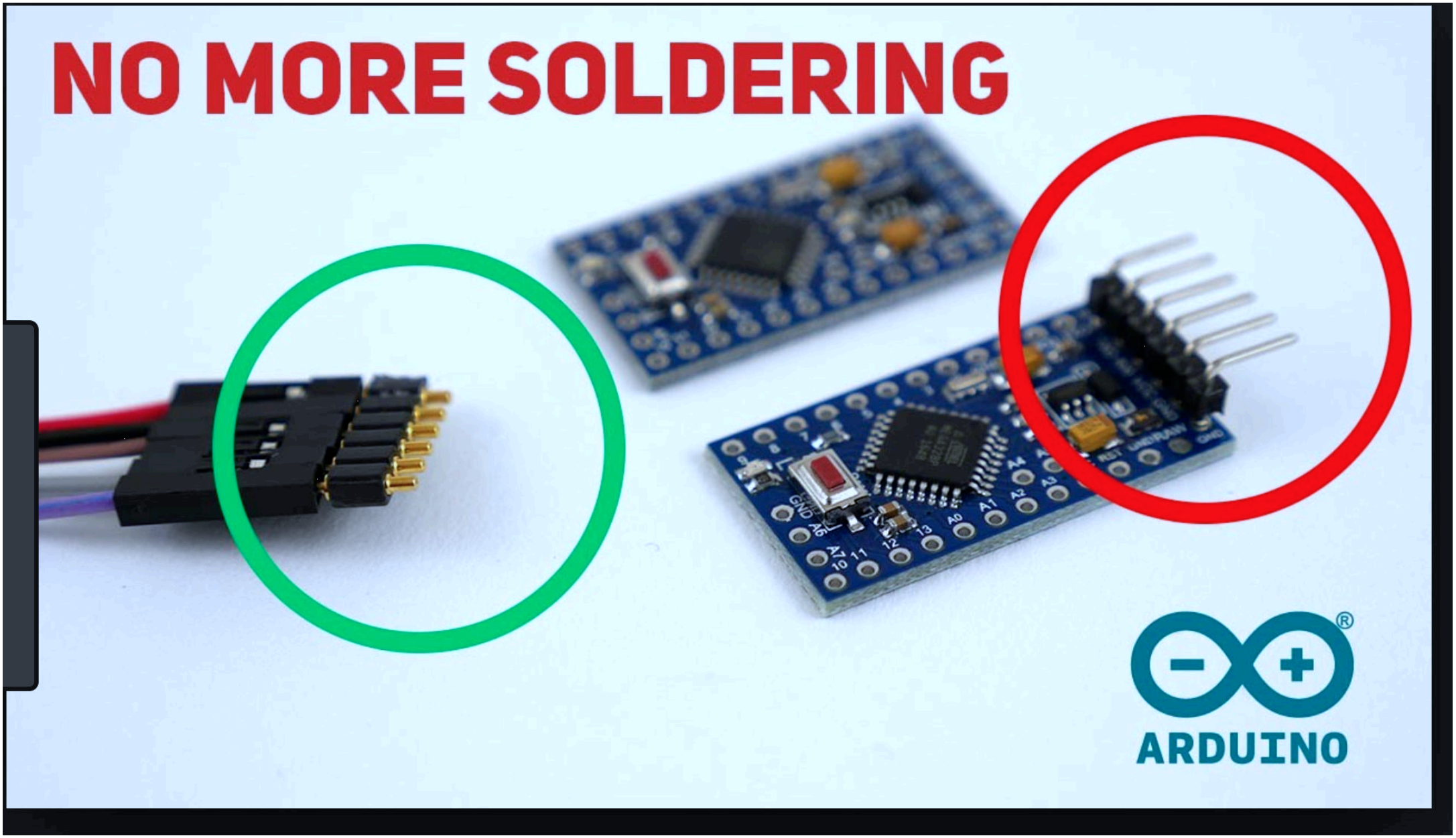
Max package power:
when powered via RAW: 150mA
when powered directly: 500mA

Max power per pin:
40.0mA absolute maximum
<=20.0mA recommended

SUM(DD-D5,A7,A6) <= 100mA IOL
SUM(B0-B5,D6-D7) <= 100mA IOL
SUM(DD-D4) <= 100mA IOL

SUM(DD-D5,A7,A6) <= 150mA IOH
SUM(B0-B5,D6-D7) <= 150mA IOH
SUM(DD-D4) <= 100mA IOH

AVR Package Pin
AVR Pin Description
Arduino Pin Identifier (digital)
Arduino Pin Identifier (analog)
PWM-Pin
Interrupt Identifier
Special Function
Power
USART Pin
SPI Pin
I²C Pin


NO MORE SOLDERING
ARDUINO

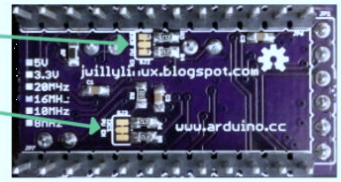
PB Pro Mini

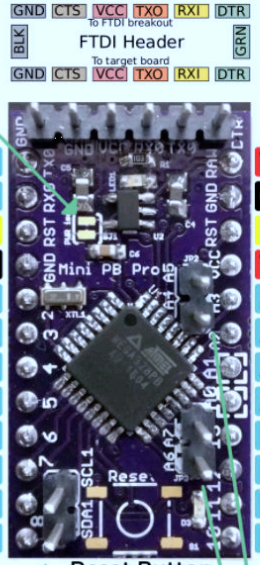
Using the ATMega328PB

1-20MHz @3.0-5.0V

Pin-compatible with the Arduino Pro Mini
(except with extra D22, D23)

Reverse side:





OC4A		PCINT17	PWM	TXD0	PD1	D1	VIN						
OC3A		PCINT16	PWM	RXD0	PD0	D0	GND						
		PCINT14			PC6	!Reset	!Reset	PC6				PCINT14	
						GND	VCC						
OC3/4B	INT0	PCINT18	PWM		PD2	D2	A3/D17	PC3		ADC3		PCINT11	
OC2B	INT1	PCINT19	PWM		PD3	D3	A2/D16	PC2		ADC2		PCINT10	
XCK0	T0	PCINT20			PD4	D4	A1/D15	PC1	SCK1	ADC1		PCINT9	
OC0B	T1	PCINT21	PWM		PD5	D5	A0/D14	PC0	MISO1	ADC0		PCINT8	
OC0A	AIN0	PCINT22	PWM		PD6	D6	D13	PB5	SCK0			PCINT5	LED
	AIN1	PCINT23			PD7	D7	D12	PB4	MISO0/RXD1			PCINT4	
CLKO	ICP1	PCINT0			PB0	D8	D11	PB3	MOSI0/TXD1	PWM		PCINT3	OC2A
OC1A		PCINT1	PWM		PB1	D9	D10	PB2	!SS0	PWM		PCINT2	OC1B
							A5/D19	PC5	SCL0	ADC5		PCINT13	
							A4/D18	PC4	SDA0	ADC4		PCINT12	
	T4	PCINT25		SCL1	PE1	D23	A7/D21	PE3	MOSI1	ADC7		PCINT27	T3
AC0	ICP4	PCINT24		SDA1	PE0	D22	A6/D20	PE2	!SS1	ADC6		PCINT26	ICP3

Power
- VIN/Raw: 3.3V-16V
- VCC: 3.3V-5.0V
- Maximum current: 150mA

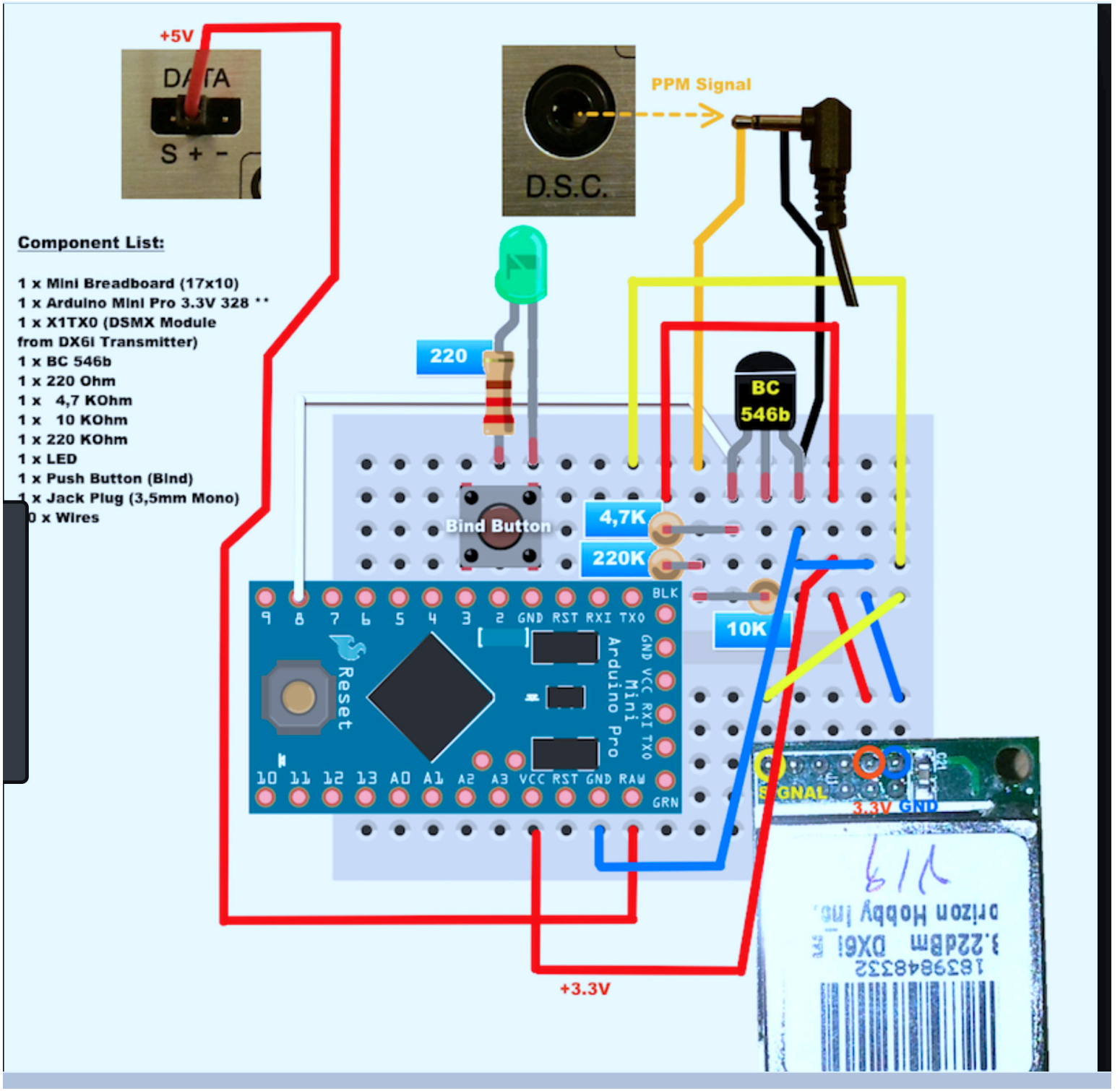
ATMega328PB
- Absolute maximum VCC: 6V
- Maximum current for chip: 100mA
- Maximum current per pin: 40mA
- 8-bit Atmel AVR core
- Flash Program Memory: 32kB
- EEPROM: 1kB
- Internal SRAM: 2kB

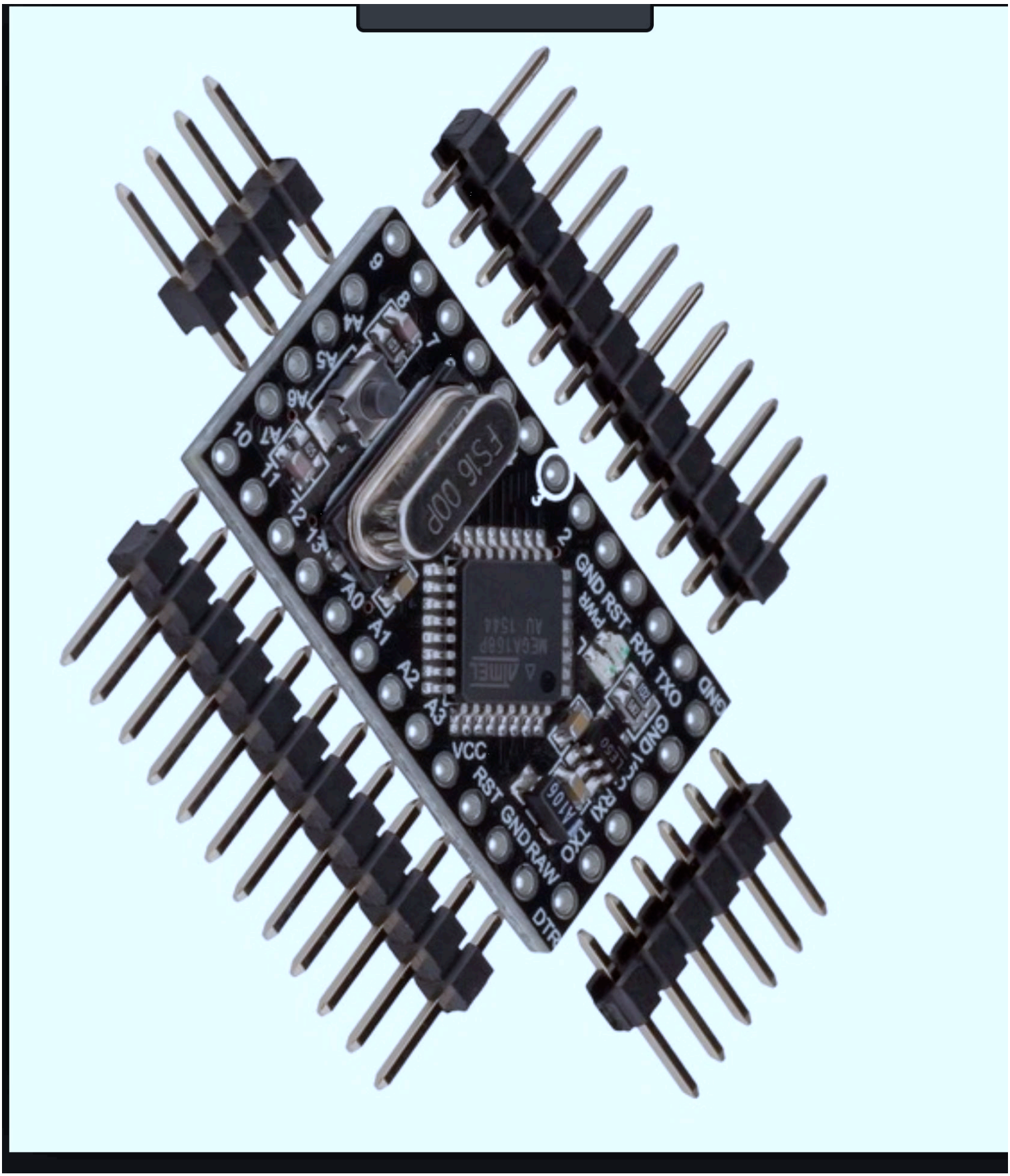
- ADC: 10-bit
- PWM: 8-bit and 16-bit
- 2 USARTs
- 2 SPIs
- 2 I2Cs
- Analog Comparator
- RTC with separate oscillator
- Peripheral Touch Controller

LEDs:
- Power: Red
- User (D13): Green

Power isolation pad
I2C pullup enable pads

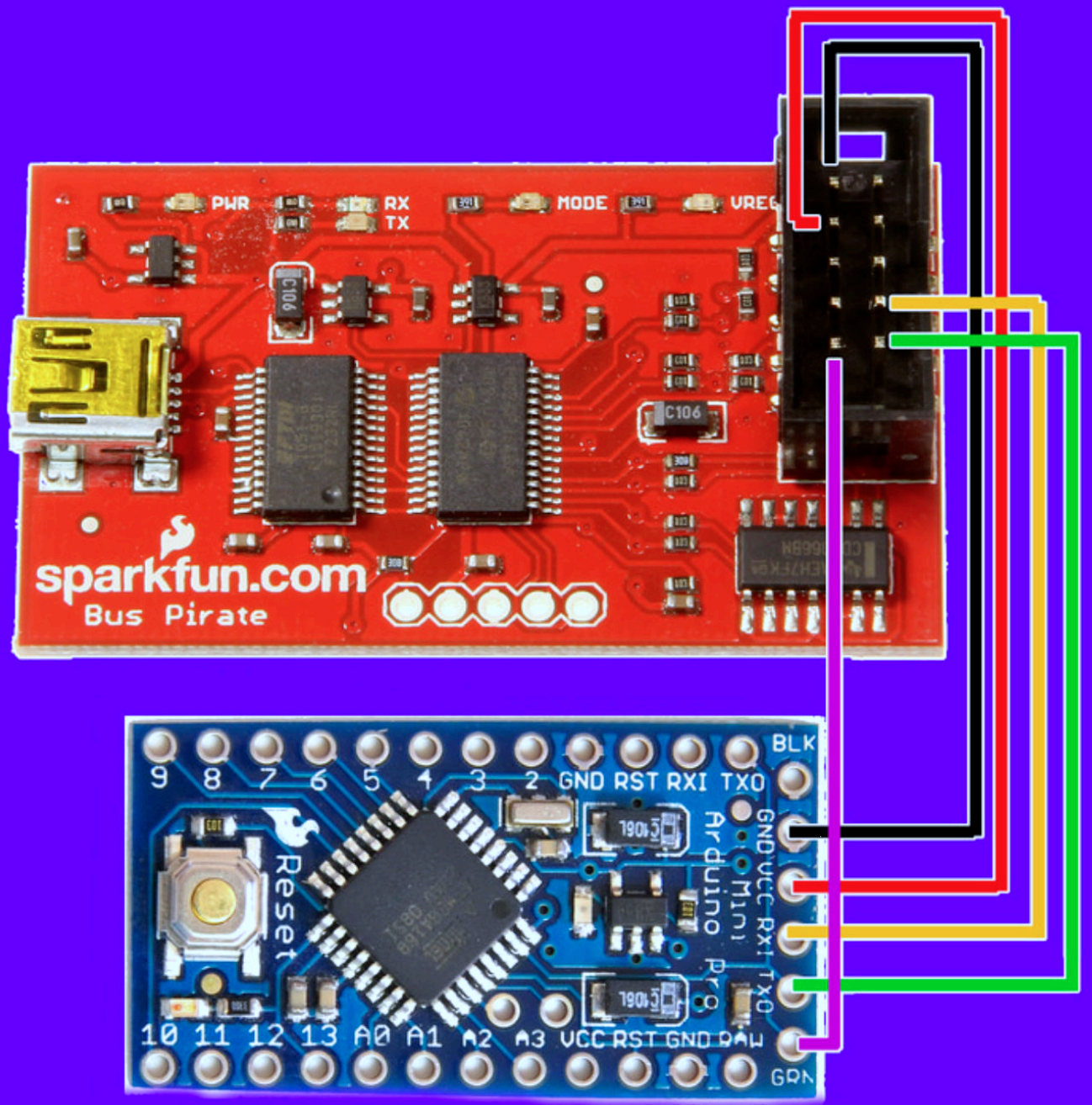
By JWilly
v1.0
Based on the Arduino Pro Mini
jwillylinux.blogspot.com
https://github.com/kwispykweems2

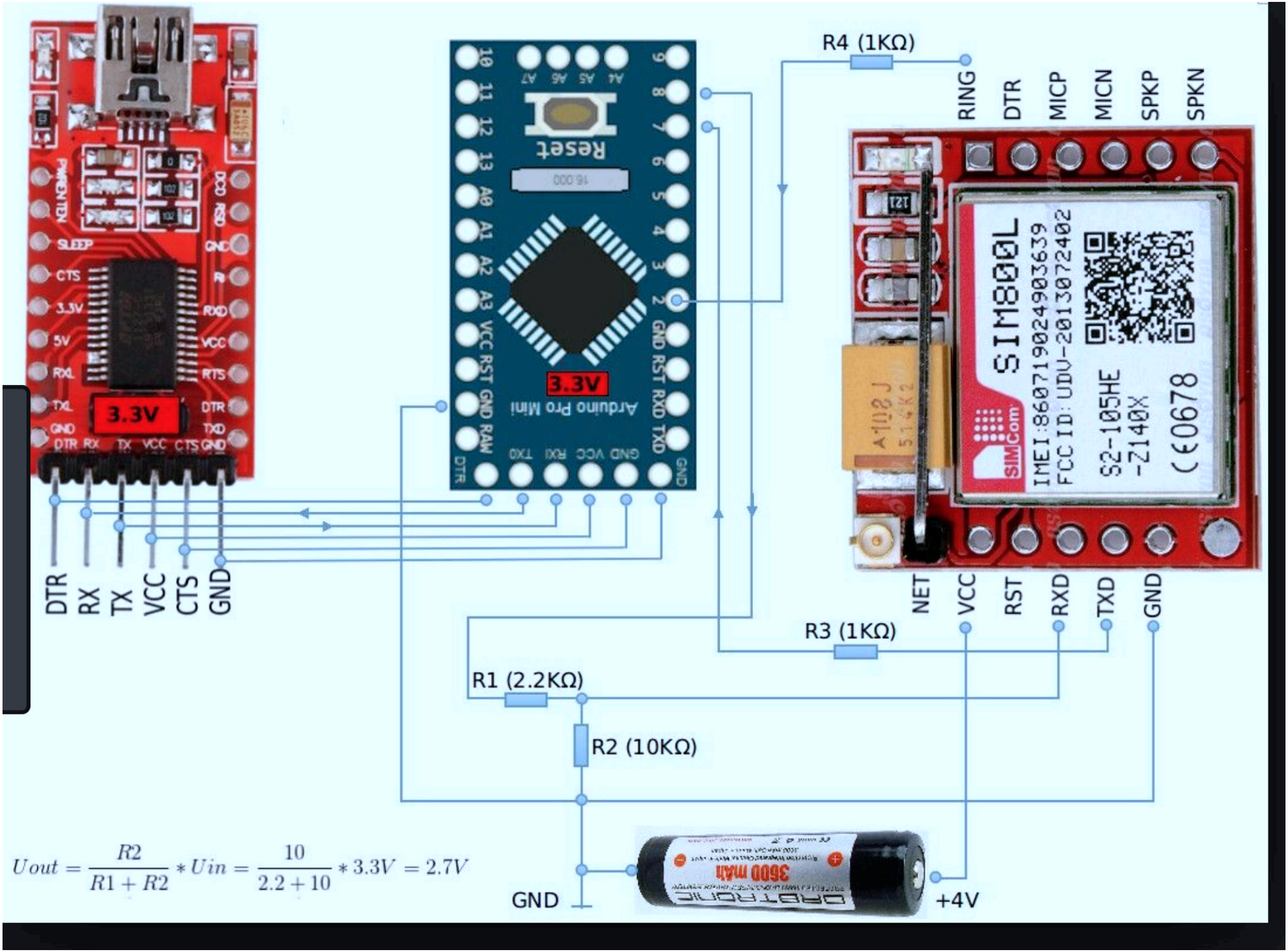

+5V
DATA
S + -
D.S.C.
PPM Signal
Component List:
1 x Mini Breadboard (17x10)
1 x Arduino Mini Pro 3.3V 328 **
1 x X1TX0 (DSMX Module
from DX6i Transmitter)
1 x BC 546b
1 x 220 Ohm
1 x 4,7 KOhm
1 x 10 KOhm
1 x 220 KOhm
1 x LED
1 x Push Button (Bind)
1 x Jack Plug (3,5mm Mono)
0 x Wires
220
BC
546b
Bind Button
4,7K
220K
10K
BLK
9 8 7 6 5 4 3 2 GND RST RXI TXO
GND VCC RXI TXO
Reset
Arduino Pro
Mini
10 11 12 13 A0 A1 A2 A3 VCC RST GND RAW
GRN
SIGNAL
3.3V GND
+3.3V

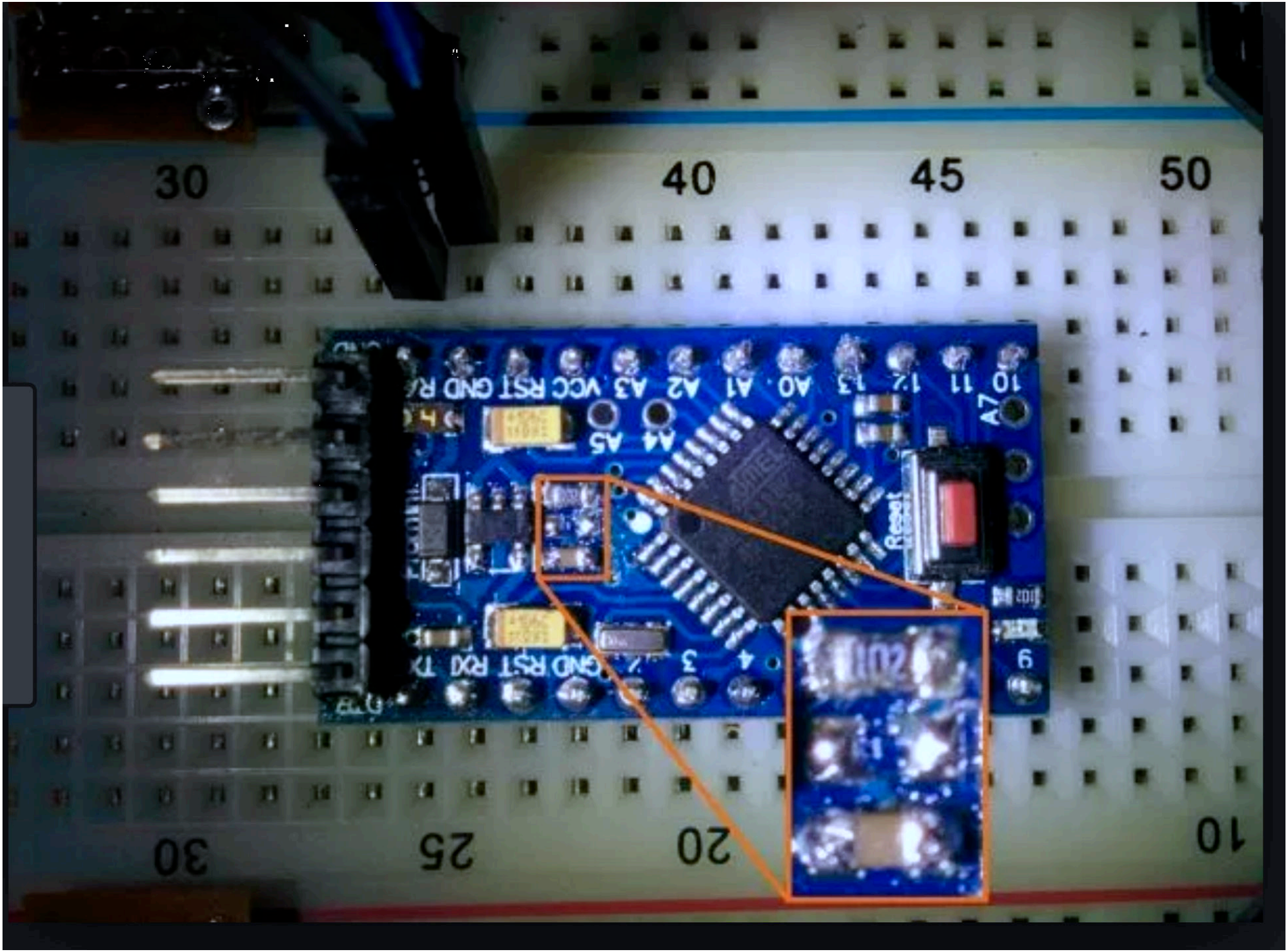

GND RST RXI TXO GND
VCC RST GNDRAW DTR
A0 A1 A2 A3
ATMEL

How to use connect
a “Bus Pirate”
as a UART to program
an Arduino Pro Mini.

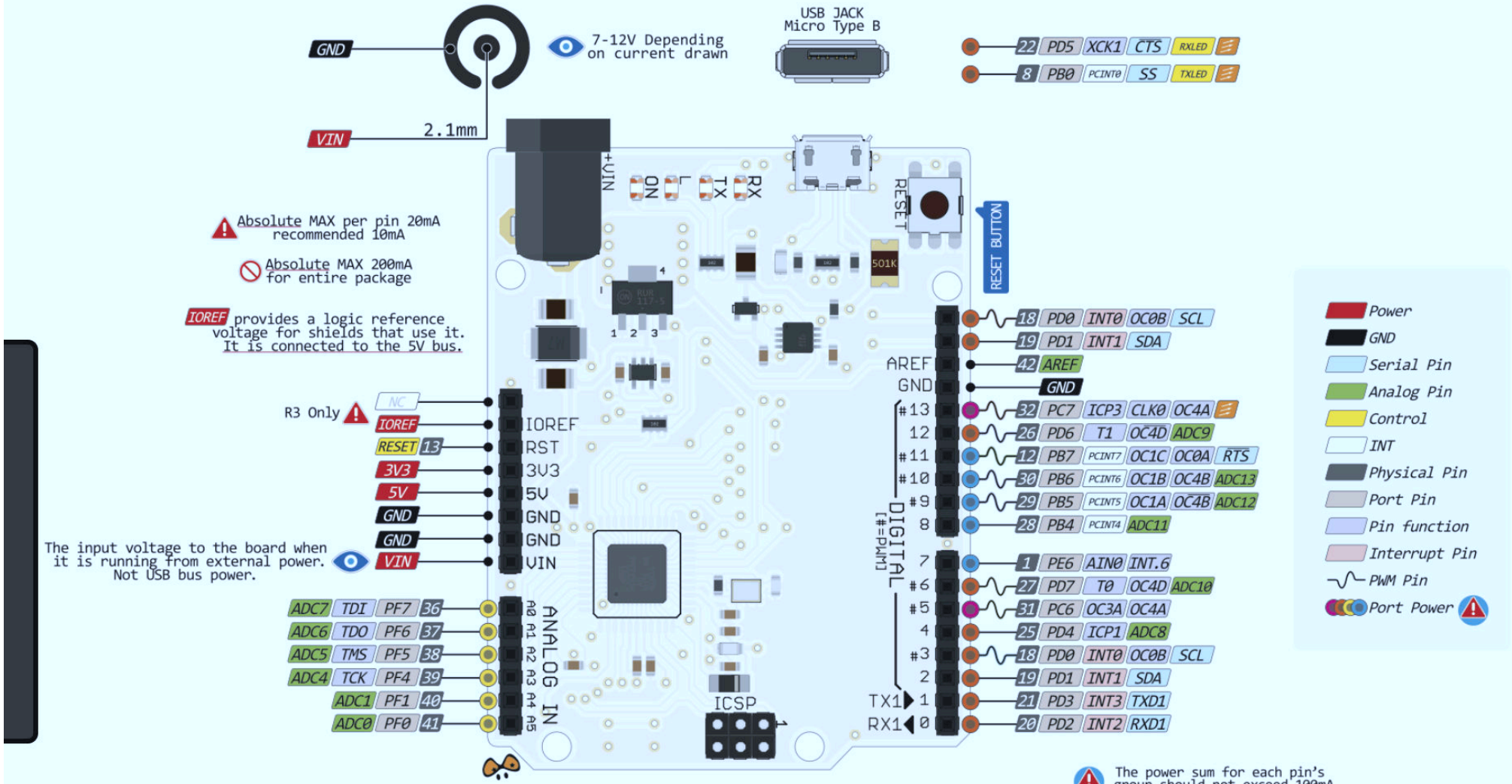
The Arduino Pro Mini
doesn’t come with
a built-in FTDI chip
(serial to USB), and
if you don’t have a serial
to USB but DO have a
“Bus Pirate”, you can
connect it to the Arduino
Pro Mini as shown,
and program it
via Arduino IDE.




R4 (1KΩ)
RING
DTR
MICP
MICN
SPKP
SPKN
SIM800L
IMEI:860719024903639
FCC ID: UDV-201307240Z
S2-105HE
-2140X
NET
VCC
RST
RXD
TXD
GND
R3 (1KΩ)
R1 (2.2KΩ)
R2 (10KΩ)
GND
+4V
3.3V
Reset
Arduino Pro Mini
DTR
RX
TX
VCC
CTS
GND
$Uout = \frac{R2}{R1+R2} * Uin = \frac{10}{2.2+10} * 3.3V = 2.7V$

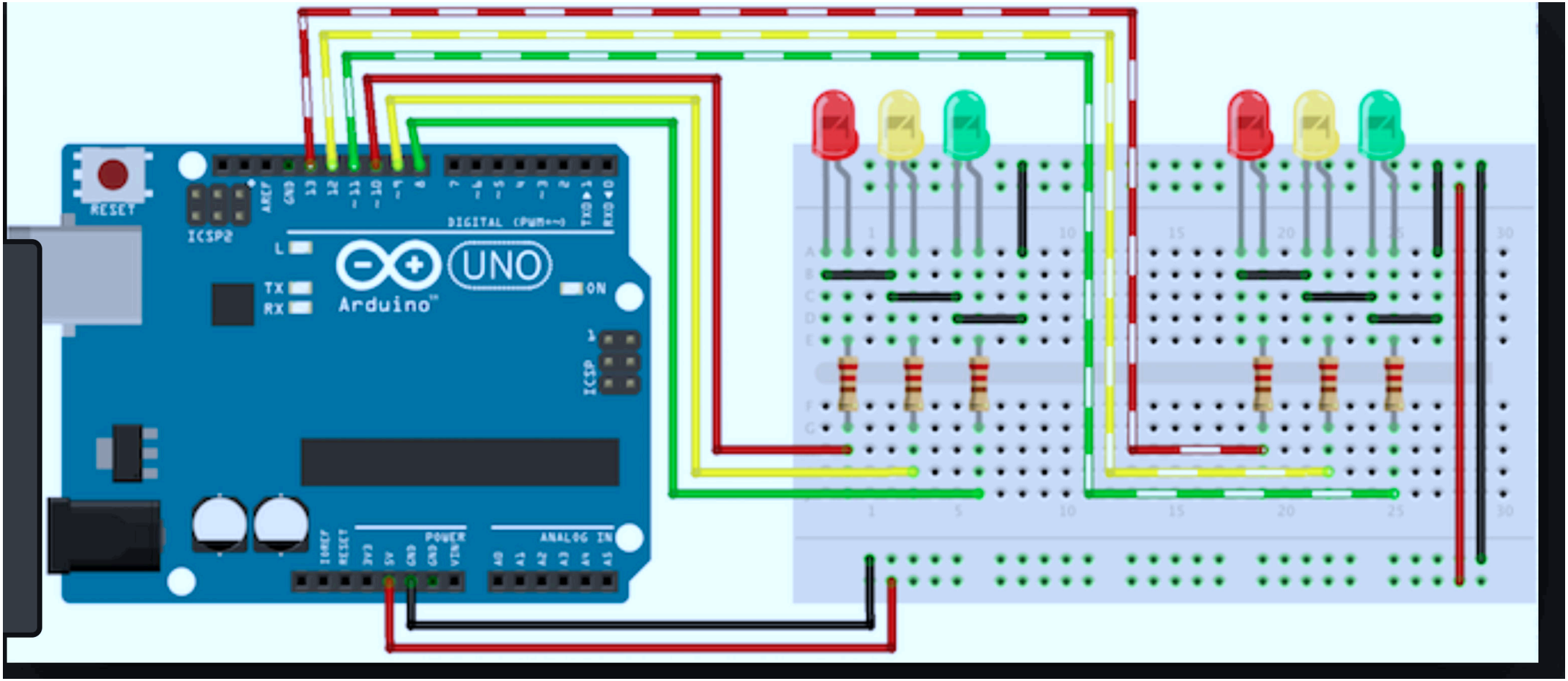

30
40
45
50
25
20
10
A7
A5
A4
A3
A2
A1
A0
RST
GND
VCC
RXD
TXD
Reset

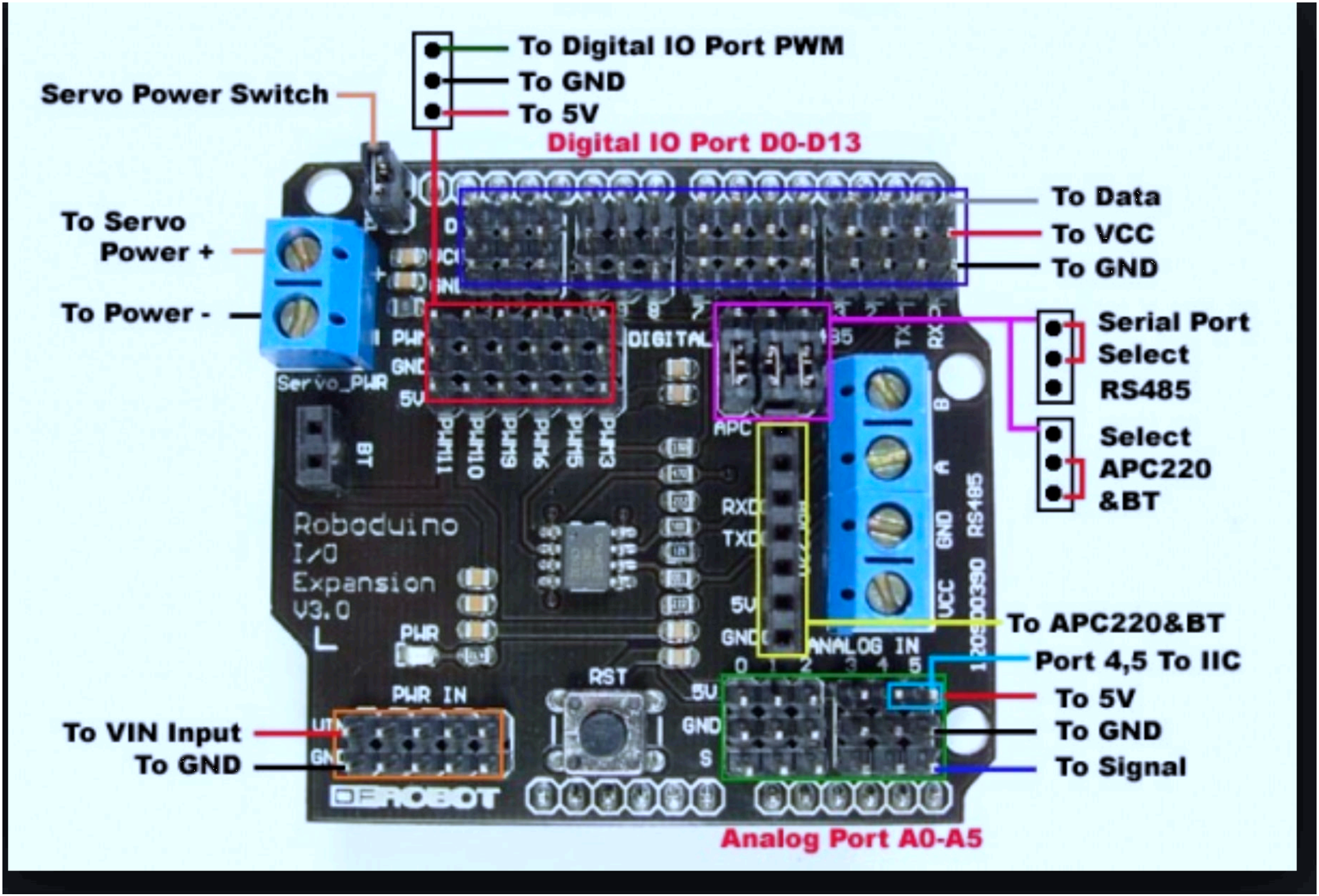
LEONARDO PINOUT

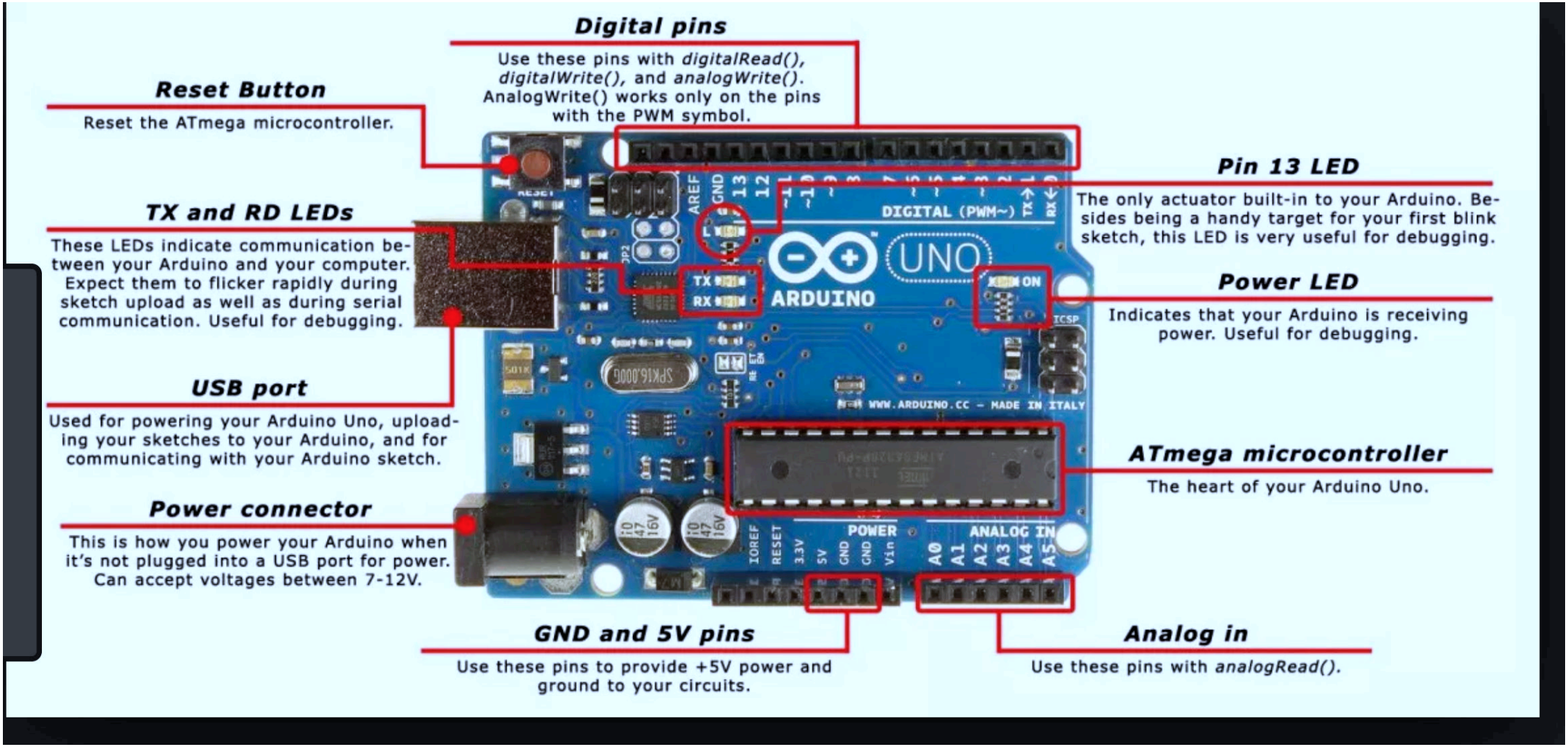


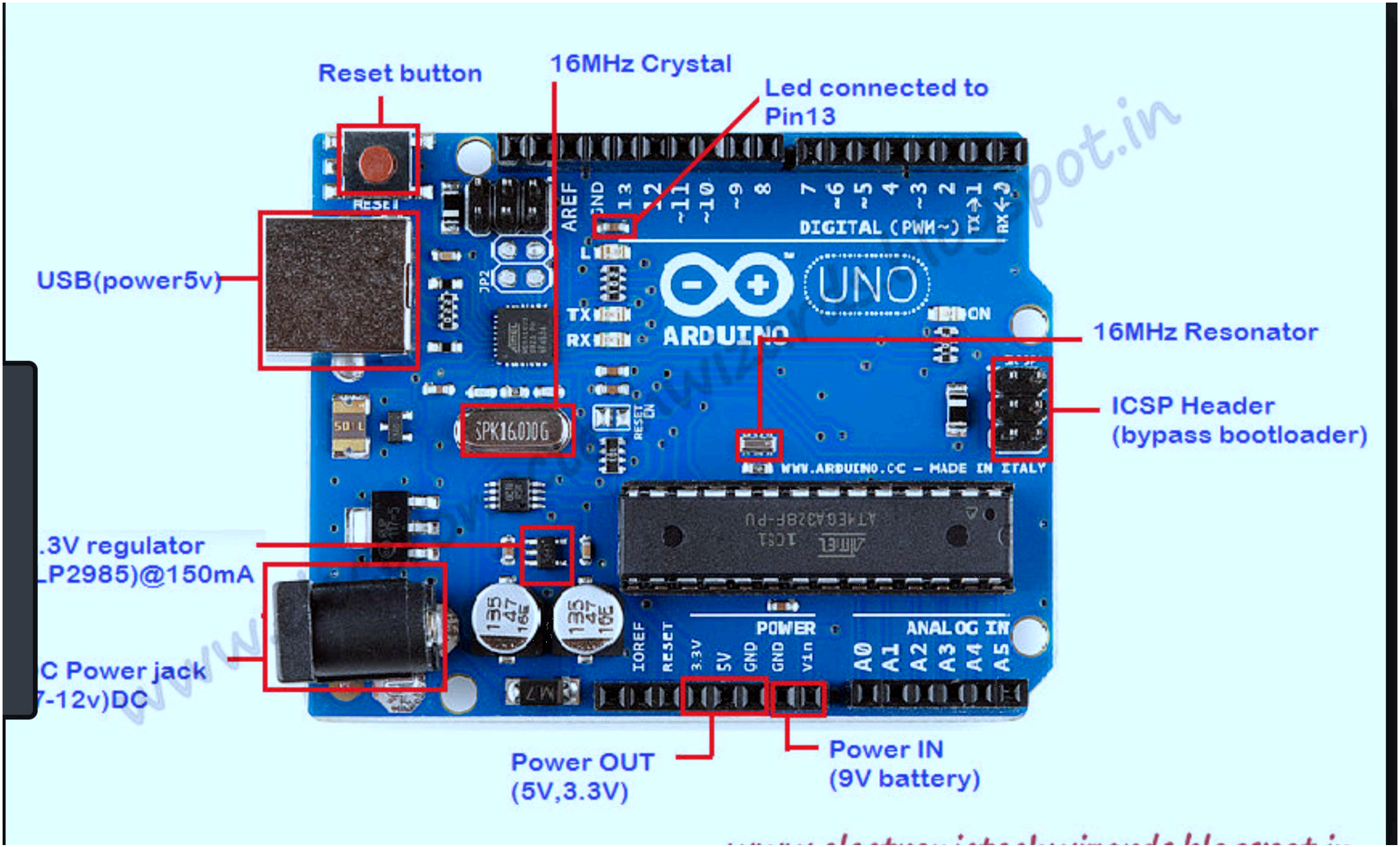
bq
www.bq.com

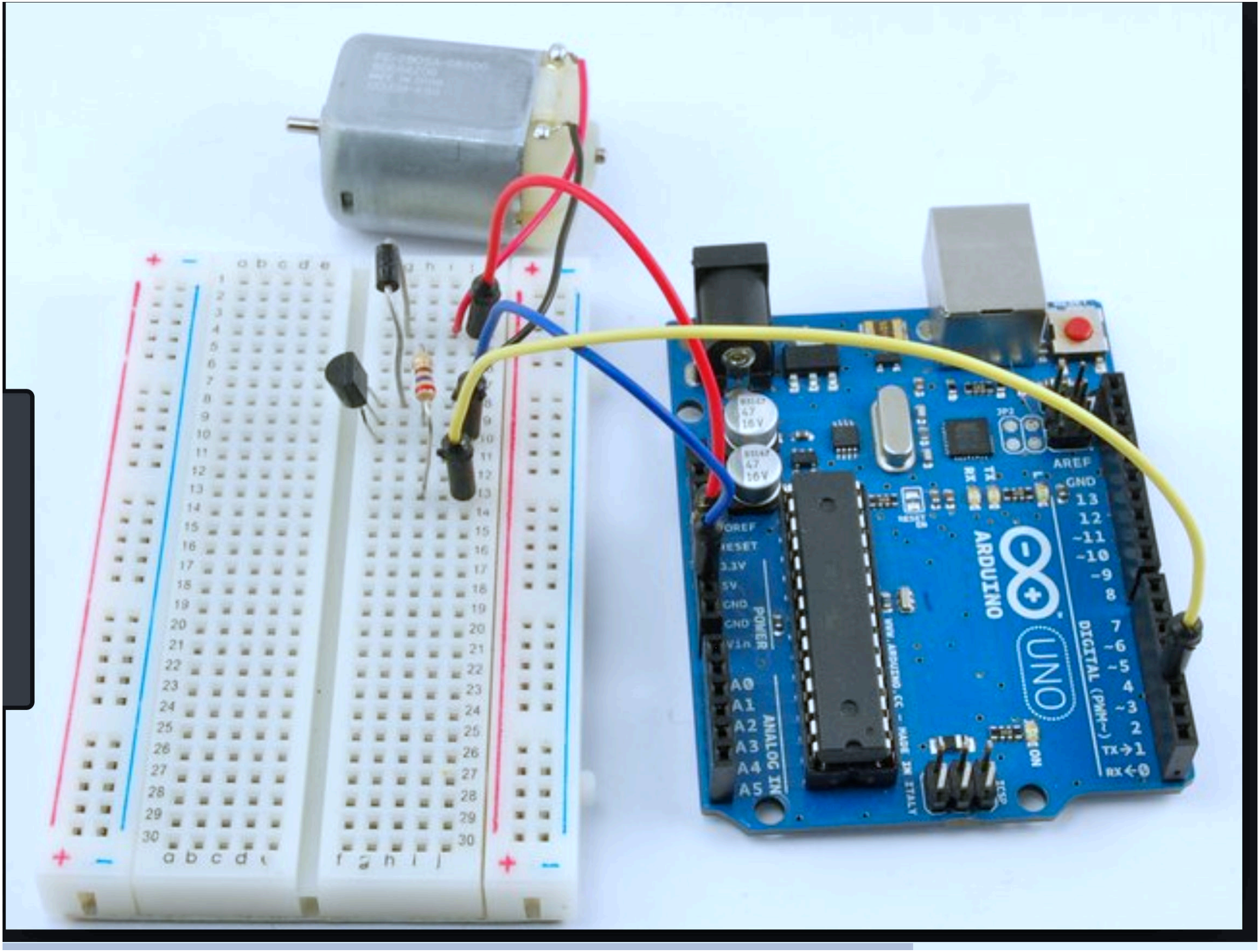
31 JUL 2014
ver 3 rev 0

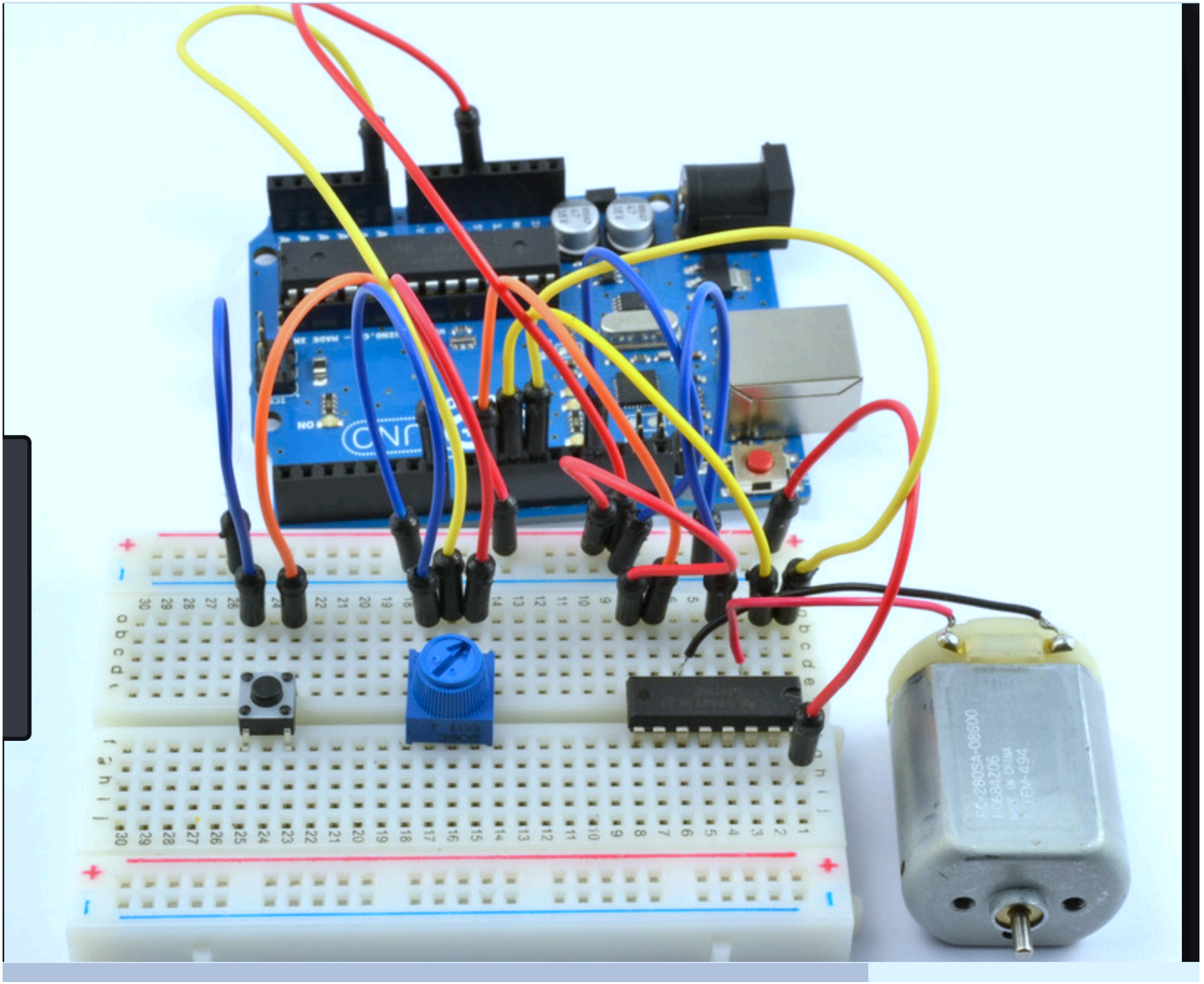

RESET
ICSP2
DIGITAL (PWM~)
L
TX
RX
UNO
Arduino
ON
ICSP
POWER
ANALOG IN

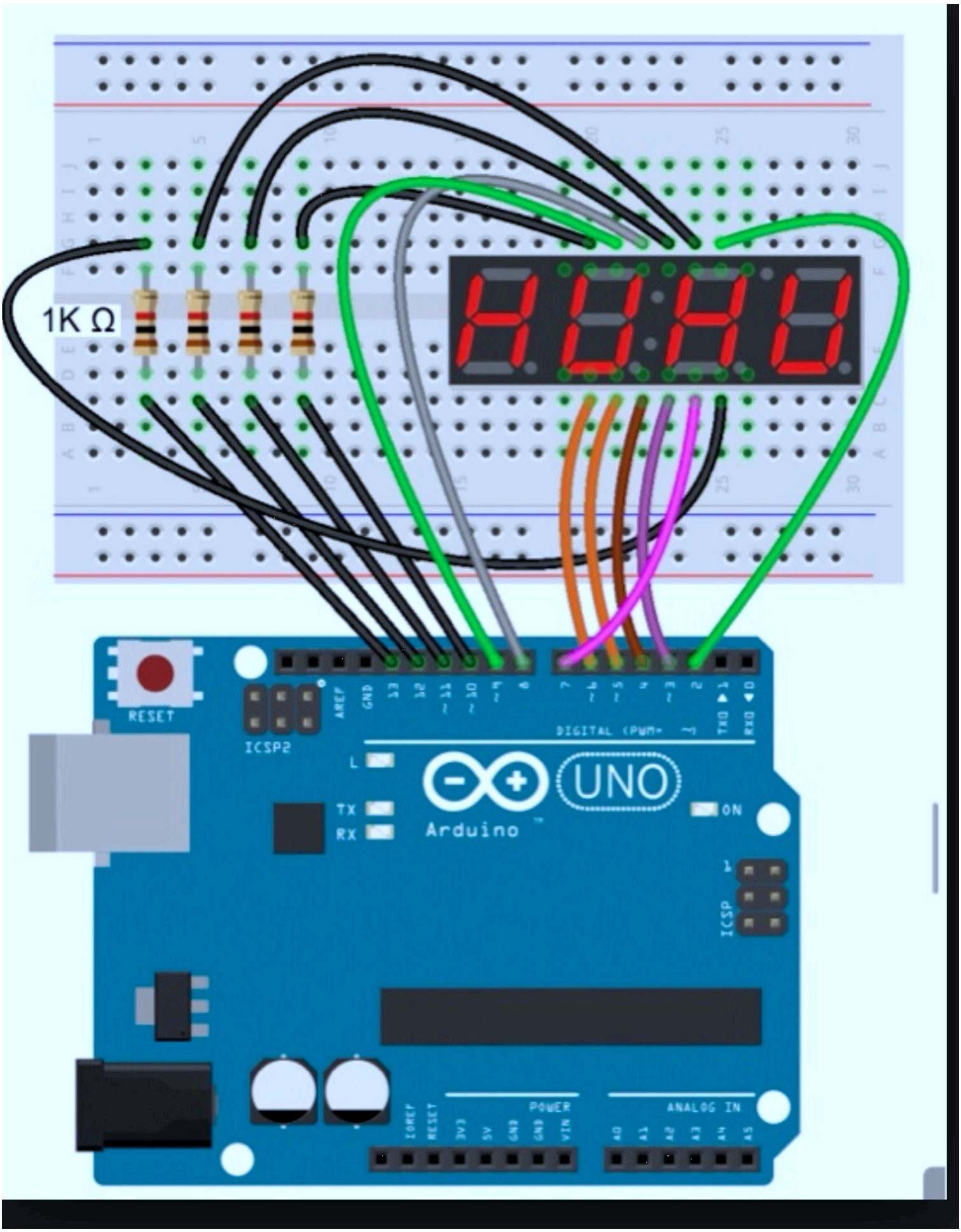

To Digital IO Port PWM
To GND
To 5V
Servo Power Switch
Digital IO Port D0-D13
To Data
To VCC
To GND
To Servo Power +
To Power -
Serial Port Select RS485
Select APC220 &BT
To APC220&BT
Port 4,5 To IIC
To 5V
To GND
To Signal
To VIN Input
To GND
Analog Port A0-A5
Roboduino I/O Expansion V3.0
DIGITAL
ANALOG IN
PWR IN
RST
Servo_PWR
APC
DFROBOT

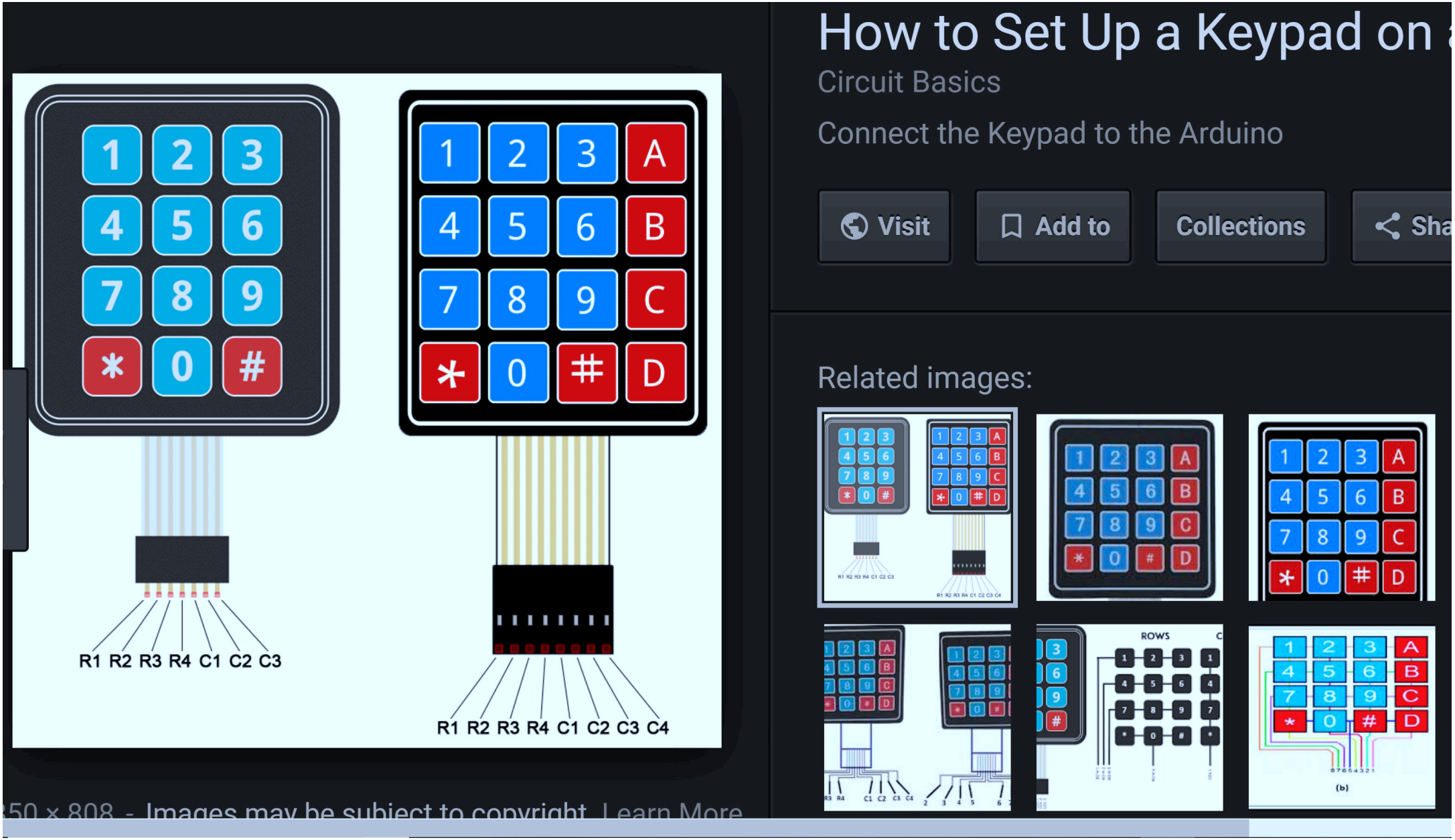

Digital pins
Use these pins with digitalRead(), digitalWrite(), and analogWrite(). AnalogWrite() works only on the pins with the PWM symbol.
Reset Button
Reset the ATmega microcontroller.
Pin 13 LED
The only actuator built-in to your Arduino. Besides being a handy target for your first blink sketch, this LED is very useful for debugging.
TX and RD LEDs
These LEDs indicate communication between your Arduino and your computer. Expect them to flicker rapidly during sketch upload as well as during serial communication. Useful for debugging.
Power LED
Indicates that your Arduino is receiving power. Useful for debugging.
USB port
Used for powering your Arduino Uno, uploading your sketches to your Arduino, and for communicating with your Arduino sketch.
ATmega microcontroller
The heart of your Arduino Uno.
Power connector
This is how you power your Arduino when it's not plugged into a USB port for power. Can accept voltages between 7-12V.
GND and 5V pins
Use these pins to provide +5V power and ground to your circuits.
Analog in
Use these pins with analogRead().
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN
WWW.ARDUINO.CC – MADE IN ITALY


Reset button
16MHz Crystal
Led connected to
Pin13
USB(power5v)
16MHz Resonator
ICSP Header
(bypass bootloader)
.3V regulator
LP2985)@150mA
C Power jack
7-12v)DC
Power OUT
(5V,3.3V)
Power IN
(9V battery)
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
WWW.ARDUINO.CC - MADE IN ITALY


ARDUINO
UNO
DIGITAL (PWM~)
ANALOG IN
POWER
AREF
GND
A0
A1
A2
A3
A4
A5




1K Ω
RESET
ICSP2
L
TX
RX
Arduino
UNO
ON
ICSP
DIGITAL (PWM~)
POWER
ANALOG IN



How to Set Up a Keypad on

Circuit Basics

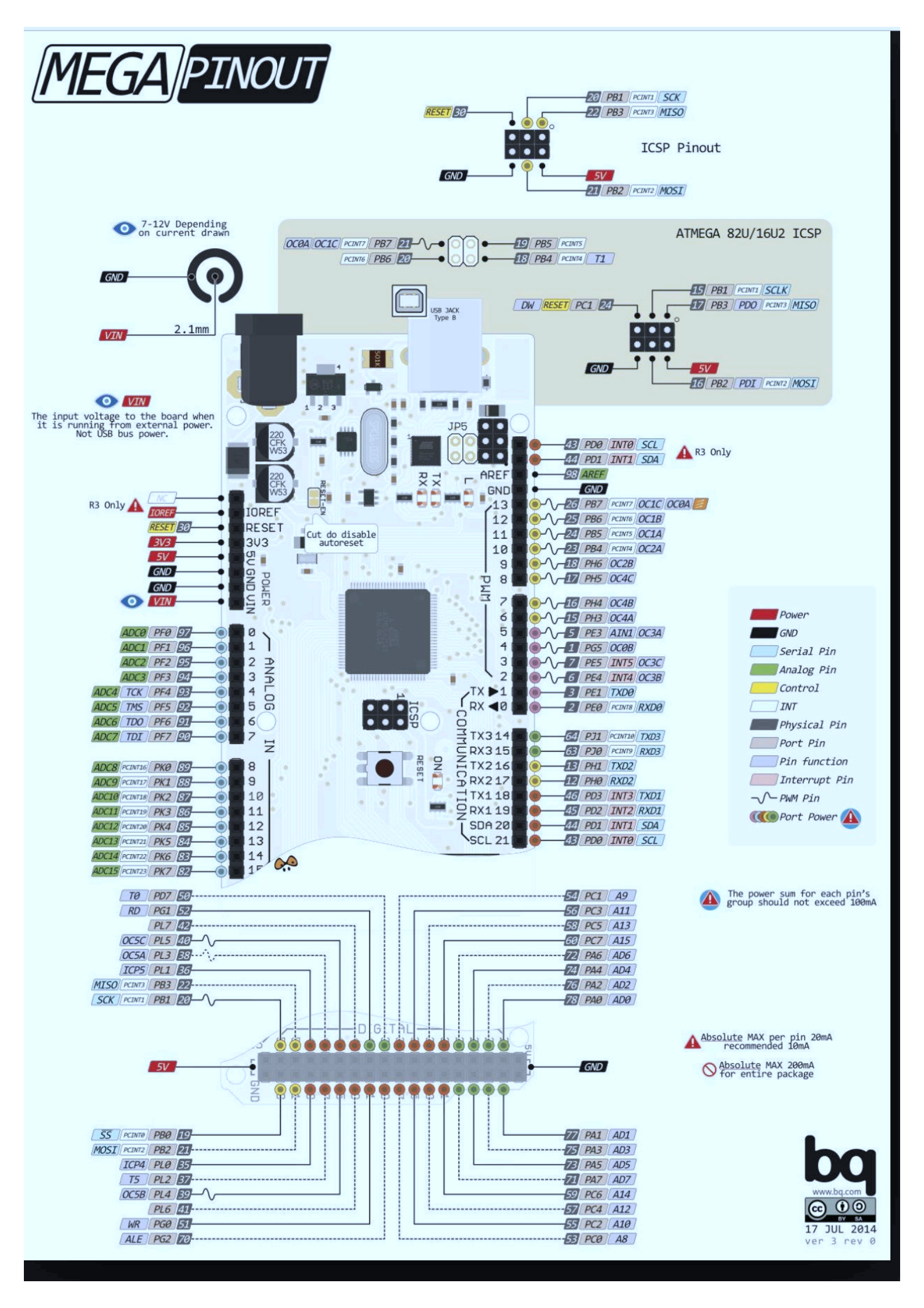
Connect the Keypad to the Arduino

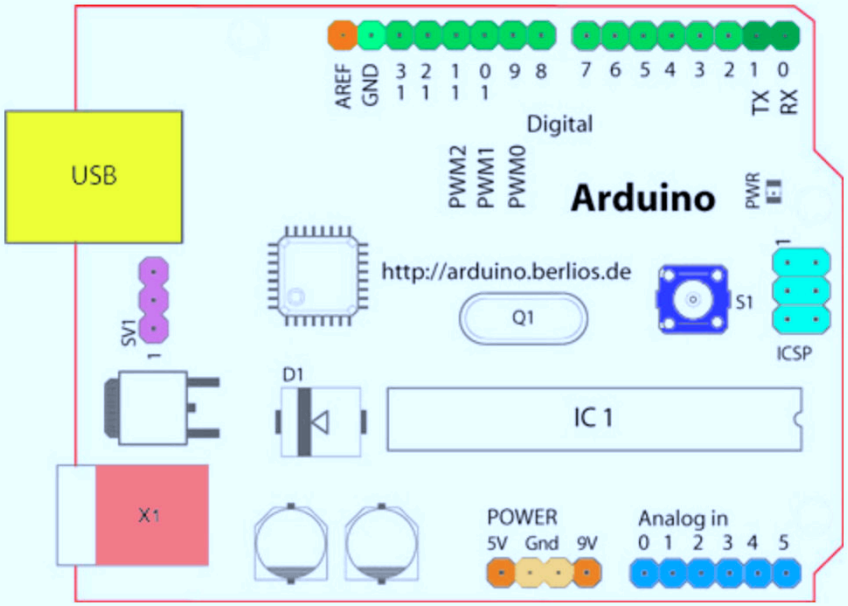
Visit | Add to | Collections | Sha

Related images:

250 × 808 - Images may be subject to copyright. Learn More

MEGA PINOUT


ICSP Pinout
RESET 30
20 PB1 PCINT1 SCK
22 PB3 PCINT3 MISO
GND
5V
21 PB2 PCINT2 MOSI
7-12V Depending on current drawn
GND
VIN
2.1mm
ATMEGA 82U/16U2 ICSP
OC0A OC1C PCINT7 PB7 21
PCINT6 PB6 20
19 PB5 PCINT5
18 PB4 PCINT4 T1
DW RESET PC1 24
15 PB1 PCINT1 SCLK
17 PB3 PDO PCINT3 MISO
GND
5V
16 PB2 PDI PCINT2 MOSI
USB JACK Type B
VIN
The input voltage to the board when it is running from external power. Not USB bus power.
220 CFK W53
220 CFK W53
JP5
R3 Only
NC
IOREF
RESET 30
3V3
5V
GND
GND
VIN
POWER
Cut do disable autoreset
AREF
GND
13
12
11
10
9
8
PWM
7
6
5
4
3
2
TX ▶1
RX ◀0
COMMUNICATION
TX3 14
RX3 15
TX2 16
RX2 17
TX1 18
RX1 19
SDA 20
SCL 21
43 PD0 INT0 SCL
44 PD1 INT1 SDA
98 AREF
GND
26 PB7 PCINT7 OC1C OC0A
25 PB6 PCINT6 OC1B
24 PB5 PCINT5 OC1A
23 PB4 PCINT4 OC2A
18 PH6 OC2B
17 PH5 OC4C
16 PH4 OC4B
15 PH3 OC4A
5 PE3 AIN1 OC3A
1 PG5 OC0B
7 PE5 INT5 OC3C
6 PE4 INT4 OC3B
3 PE1 TXD0
2 PE0 PCINT8 RXD0
64 PJ1 PCINT10 TXD3
63 PJ0 PCINT9 RXD3
13 PH1 TXD2
12 PH0 RXD2
46 PD3 INT3 TXD1
45 PD2 INT2 RXD1
44 PD1 INT1 SDA
43 PD0 INT0 SCL
ADC0 PF0 97
ADC1 PF1 96
ADC2 PF2 95
ADC3 PF3 94
ADC4 TCK PF4 93
ADC5 TMS PF5 92
ADC6 TDO PF6 91
ADC7 TDI PF7 90
ADC8 PCINT16 PK0 89
ADC9 PCINT17 PK1 88
ADC10 PCINT18 PK2 87
ADC11 PCINT19 PK3 86
ADC12 PCINT20 PK4 85
ADC13 PCINT21 PK5 84
ADC14 PCINT22 PK6 83
ADC15 PCINT23 PK7 82
ANALOG IN
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
ICSP
RESET
ON
Power
GND
Serial Pin
Analog Pin
Control
INT
Physical Pin
Port Pin
Pin function
Interrupt Pin
PWM Pin
Port Power
R3 Only
The power sum for each pin's group should not exceed 100mA
T0 PD7 50
RD PG1 52
PL7 42
OC5C PL5 40
OC5A PL3 38
ICP5 PL1 36
MISO PCINT3 PB3 22
SCK PCINT1 PB1 20
54 PC1 A9
56 PC3 A11
58 PC5 A13
60 PC7 A15
72 PA6 AD6
74 PA4 AD4
76 PA2 AD2
78 PA0 AD0
DIGITAL
5V
GND
GND
Absolute MAX per pin 20mA recommended 10mA
Absolute MAX 200mA for entire package
SS PCINT0 PB0 19
MOSI PCINT2 PB2 21
ICP4 PL0 35
T5 PL2 37
OC5B PL4 39
PL6 41
WR PG0 51
ALE PG2 70
77 PA1 AD1
75 PA3 AD3
73 PA5 AD5
71 PA7 AD7
59 PC6 A14
57 PC4 A12
55 PC2 A10
53 PC0 A8
bq
www.bq.com
17 JUL 2014
ver 3 rev 0



Starting clockwise from the top center:

- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - *These pins cannot be used for digital i/o (***digitalRead** *and* **digitalWrite***) if you are also using serial communication (e.g.* **Serial.begin***).*
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

Digital Pins

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pinMode(), digitalRead(), and digitalWrite() commands. Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
- **BT Reset: 7.** (Arduino BT-only) Connected to the reset line of the bluetooth module.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED: 13.** On the Diecimila and LilyPad, there is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

Analog Pins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

- **I2C: 4 (SDA) and 5 (SCL).** Support I2C (TWI) communication using the Wire library (documentation on the Wiring website).

Power Pins

- **VIN** (sometimes labelled "9V"). The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the LilyPad has no VIN pin and accepts only a regulated input.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** (Diecimila-only) A 3.3 volt supply generated by the on-board FTDI chip.
- **GND.** Ground pins.

Other Pins

- **AREF.** Reference voltage for the analog inputs. Not currently supported by the Arduino software.
- **Reset.** (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Planning Your Arduino Project

https://www.circuito.io/blog/arduino-debugging

Design

Come up with an idea of what you want to make, what is it going to do and how

STEP 01

Build

STEP 02

Select the components you chose on circuito.io and follow the wiring guide

Test

Check that your hardware is connected properly with the test code from circuito.io

STEP 03

Code

STEP 04

Program your circuit to do exactly what you want

Debug

Use the serial monitor, simulators and other debugging tools for troubleshooting

STEP 05

circuito.io

As we have seen, the Arduino Platform was meant for students in the beginning. However, now it is actively used by electronics makers, enthusiasts, and hobbyists all around the world to make interactive stuff. It is also used by artists. If you are studying Computer Science or Electronics, there is pretty good chance that you've seen one of the Arduino boards in Action.

We can find more information on the Arduino platform on its website https://www.arduino.cc

Features of Arduino

Arduino is the most preferred platform for the makers now-a-days because of the following features:

- **Inexpensive** – Arduino is inexpensive board. It costs less than the contemporary microcontroller trainer platforms. You can even assemble your own Arduino. The Arduino clones cost even lesser than the Official Arduino boards.
- **Cross Platform** – The official Arduino IDE is supported on Windows, Linux, Mac OSX.
- **Open Source Hardware** – The diagrams of all the Arduino boards are published under **Creative Commons License** and they are open-source.
- **Open Source Software** – Arduino can be programmed with the official Arduino IDE and AVR C Programming.

Arduino Boards and Ecosystem

Till now, we've learned what microcontroller is and also learned that most of the major Arduino boards use AVR microcontrollers. A few also use ARM microprocessors. In this section, we will understand what Arduino ecosystem is and have a look at few major member boards of the Arduino Ecosystem.

Arduino has got a very vibrant ecosystem with plethora of products. These boards and products are grouped into various categories. Let's have a look at each and every category one by one.

Official Arduino Boards

Official Arduino boards carry Arduino brand on them. They are directly supported by the official Arduino IDE. They are licensed to bear Arduino Logo on them. Also, they are manufactured by authorized manufacturers.

The authorized manufacturers pay a royalty for each board which contributes towards keeping Arduino brand running. They sell the boards through the worldwide network of authorized distributors so in case of the defective boards, the buyers get the replacement and support officially.

Currently the official manufacturers are:

1. SmartProjects in Italy (http://www.arduinosrl.it)
2. Sparkfun in USA (https://www.sparkfun.com)
3. DogHunter in China (http://www.doghunter.org)

We can find the exhaustive list of the official Arduino boards here https://www.arduino.cc/en/Main/Products

Let's have a look at few of the most important of them.

Arduino Uno is the best board for those who are just getting started with Arduino platform for the first time. It is the most documented and widely used board. It uses ATmega328P microcontroller. Following is an image of Arduino UNO REV 3,

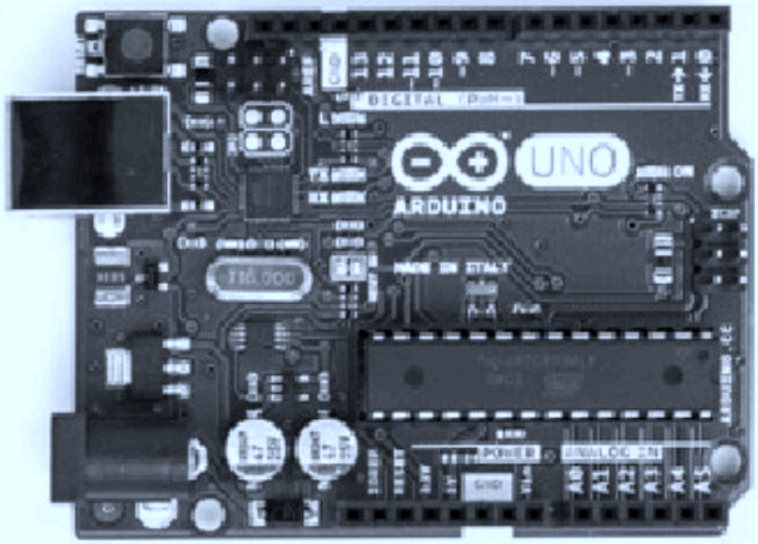


Fig. 1.2: Arduino UNO REV 3

Arduino Leonardo is another entry level board which uses ATmega32u4 microcontroller. The following is an image of Arduino Leonardo with Headers.

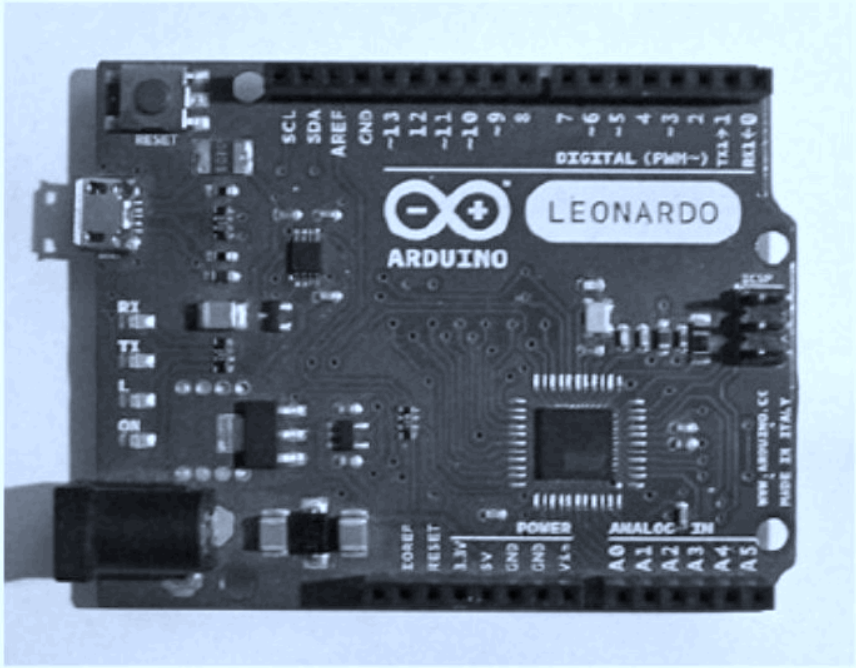


Fig. 1.3: Arduino Leonardo with Headers

Next board in the line is Arduino 101 which has 32-bit Intel Curie microcontroller. The following is an image of Arduino 101.

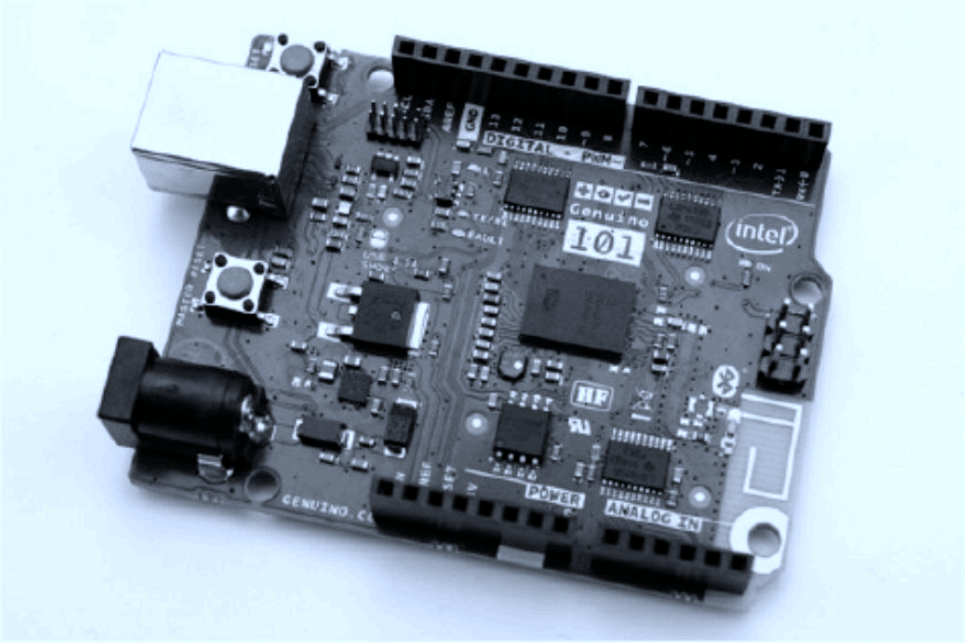


Fig. 1.4: Arduino/Genuino 101

Note: Genuino is a trademark owned by Arduino.

The Arduino Esplora is an Arduino Leonardo based board with integrated sensors and actuators. It uses ATmega32u4 microcontroller. Following image depicts an Arduino ESPLORA board.

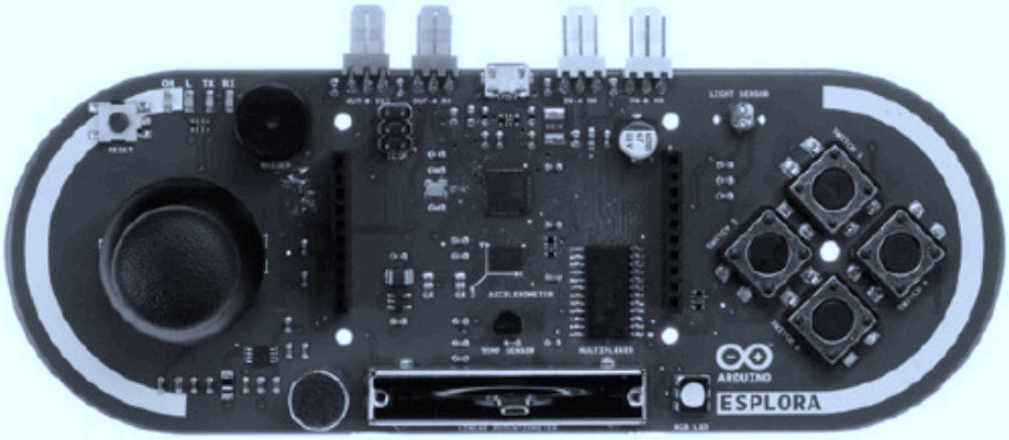


Fig. 1.5: Arduino Esplora

Arduino Micro is the smallest board of the family, used for interactive computing. The Micro is based on the ATmega32U4 microcontroller. It features a built-in USB for connection with computer.

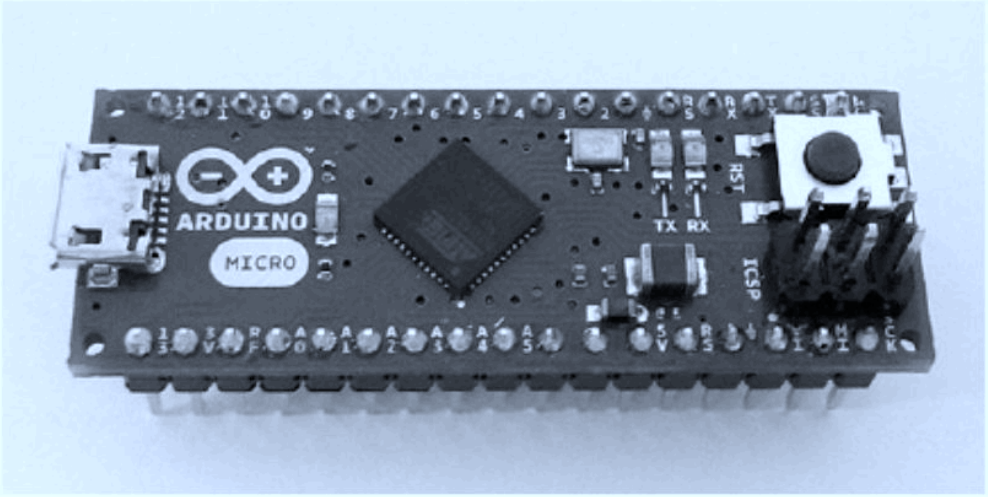


Fig. 1.6: Arduino Micro

The next member of Arduino family is Arduino Nano. It is a breadboard friendly board based on ATmega328. The following is an image of an Arduino Nano.

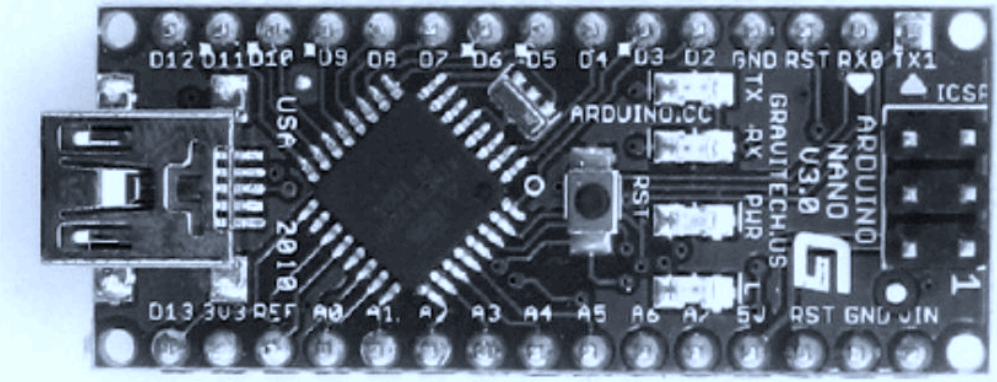


Fig. 1.7: Arduino Nano

The boards we've seen till now are the board made for the entry level users. Let's now see a more advanced line of boards with more functionality.

Arduino Mega is based on ATmega2560 microcontroller. It gives more I/O pins for use. The following is an image of Arduino Mega 2560 Rev 3.

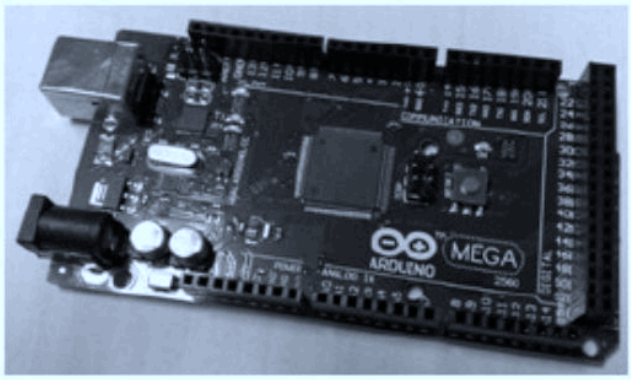


Fig. 1.8: Arduino Mega 2580 Rev 3

Arduino Zero provides 32-bit extension to the platform established by Arduino UNO R3. It uses ATSAMD21G18 microcontroller.

Fig. 1.9: Arduino Zero

Another board based on ATSAMD21G18 microcontroller is Arduino M0 PRO.

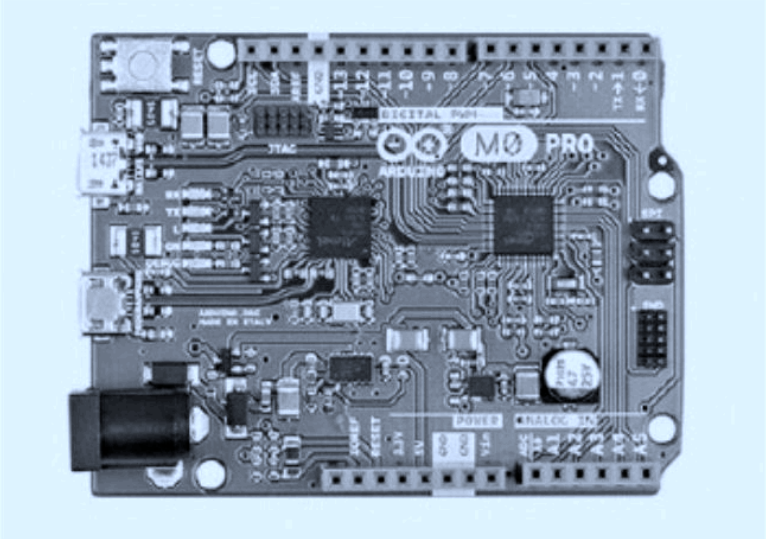


Fig. 1.10: Arduino M0 PRO

Now it's time to know about a few Linux based boards which are exclusively used for IoT. The first member is Arduino Yún. It features Atheros AR9331 microprocessor.

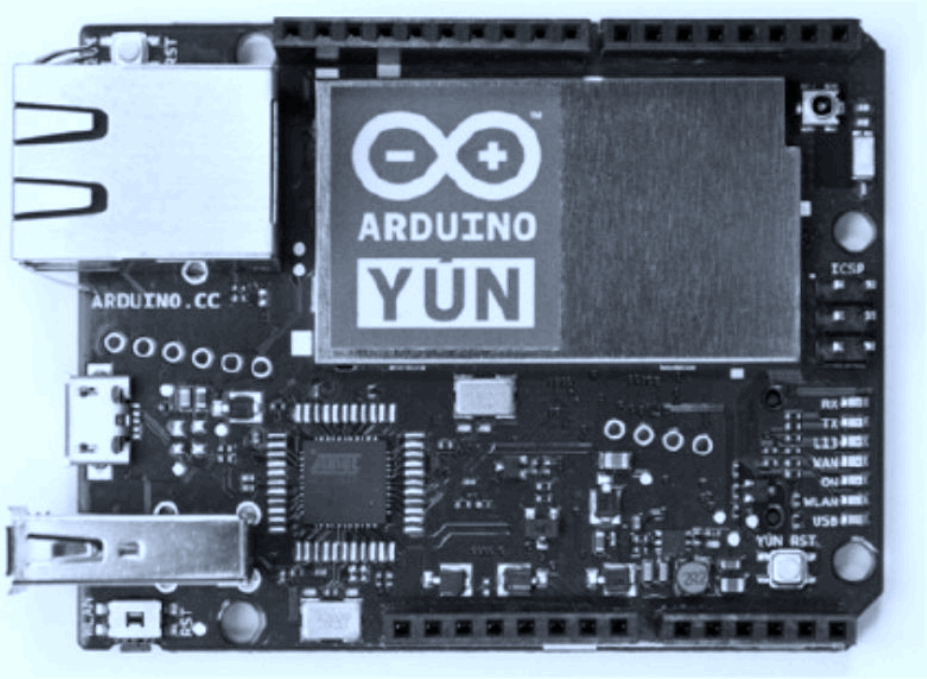


Fig. 1.11: Arduino Yún

Arduino Industrial 101 is Arduino Yún designed with small form factor.

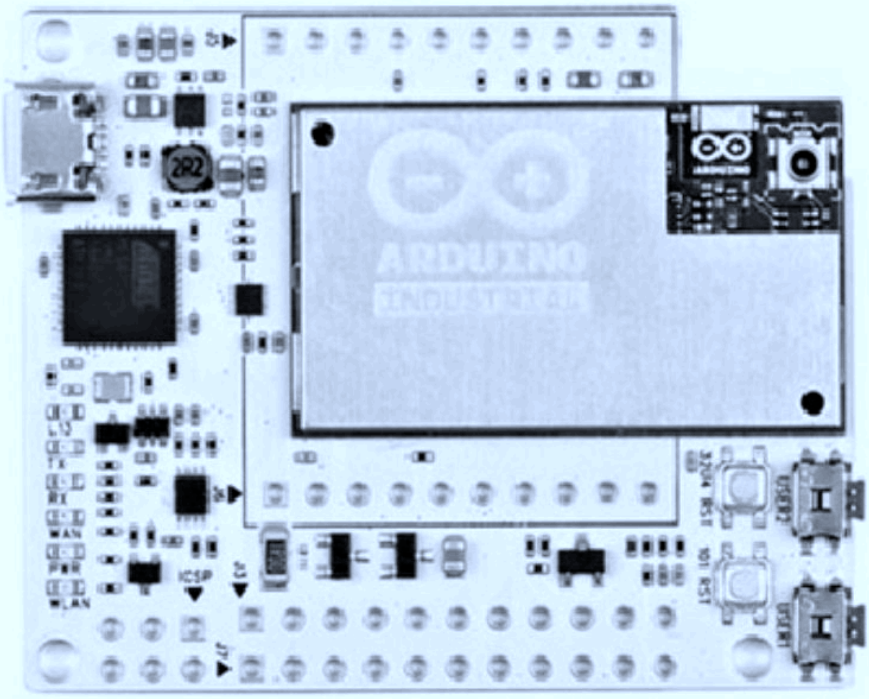


Fig 1.12: Arduino Industrial 101

Arduino Tian features a more powerful microprocessor Atheros AR9342 which is faster than Atheros 9331.

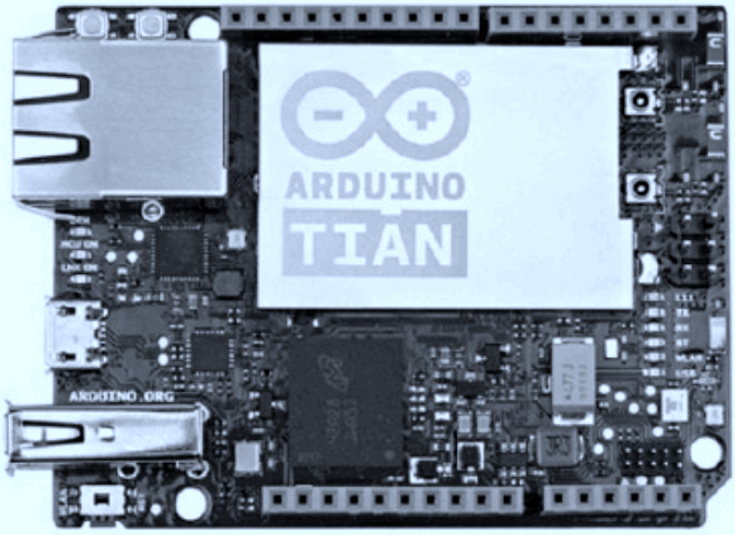


Fig. 1.13: Arduino Tian

Till now, we have seen the Arduino boards which could be used in the projects. Now, we will get introduced to a special category of miniature boards which are used for wearable projects and e-textiles. The first member is Lilypad Arduino USB.

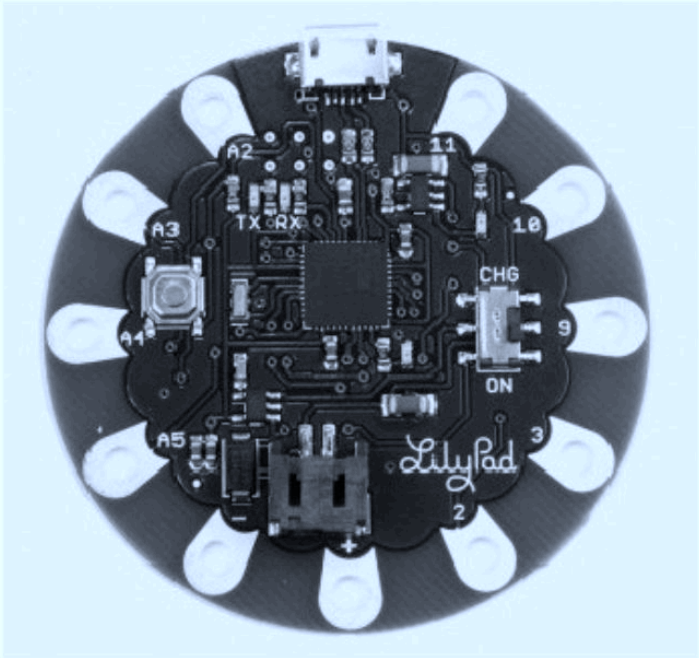


Fig. 1.14: Lilypad Arduino USB

Lilypad Arduino Mainboard uses ATmega168V or ATmega328V which are the low power versions of ATmega168 or ATmega328.

Fig. 2.6: 12V DC power supply

Alternately we can use a 9V battery with a DC barrel Jack male adapter and battery connector.

Fig. 2.7: 9V battery

Fig. 2.8: Battery connector

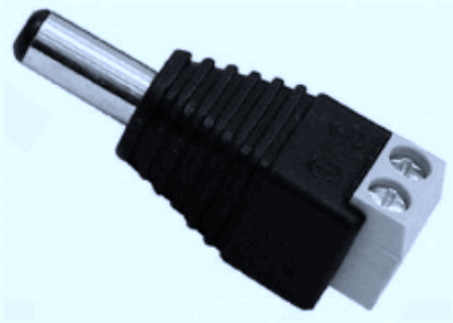


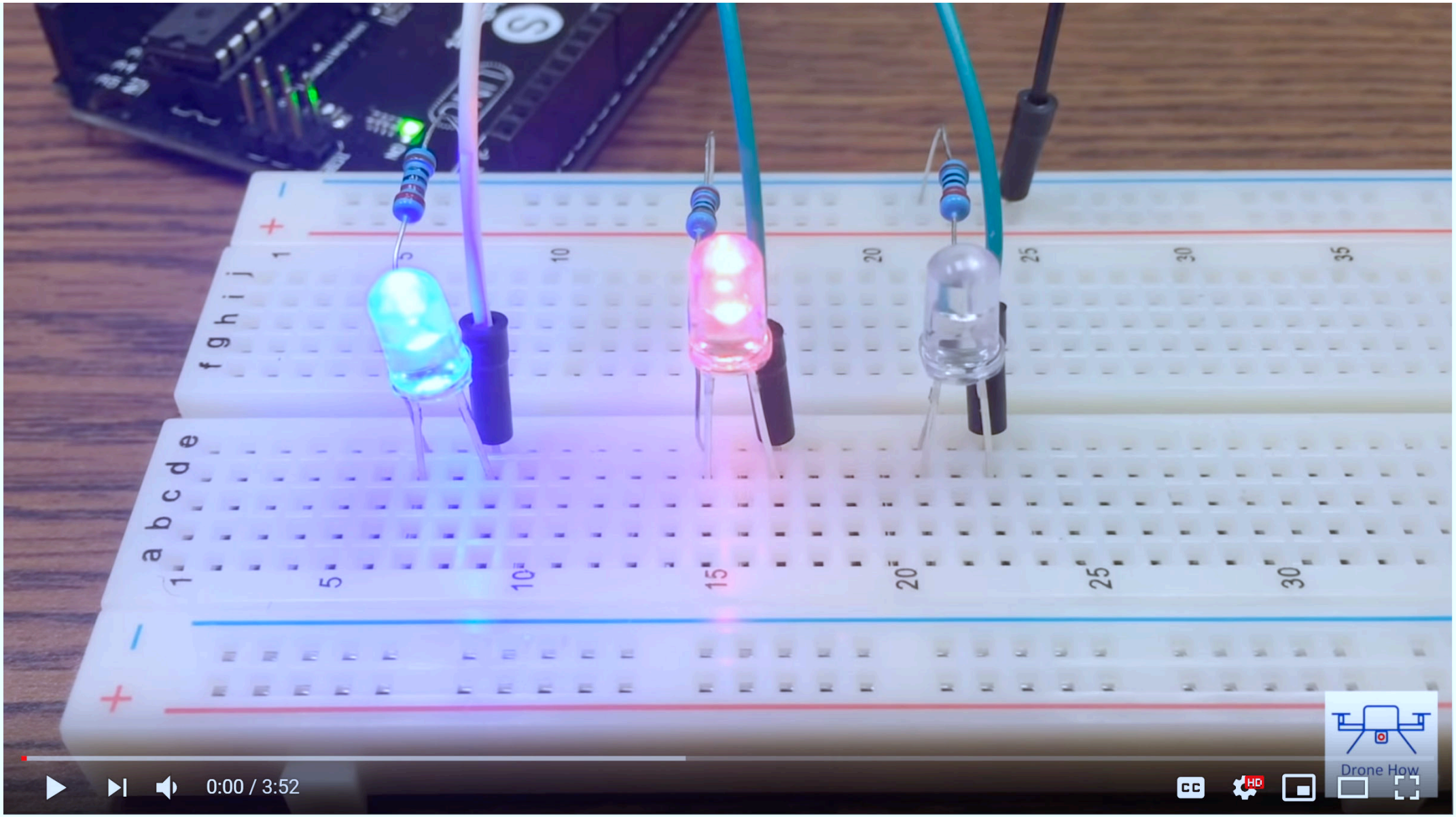
Fig. 2.9: DC Barrel Jack Adapter (male)

Note: You can find all these power supply accessories at any local electronic store or at the online e-stores like Amazon/eBay.

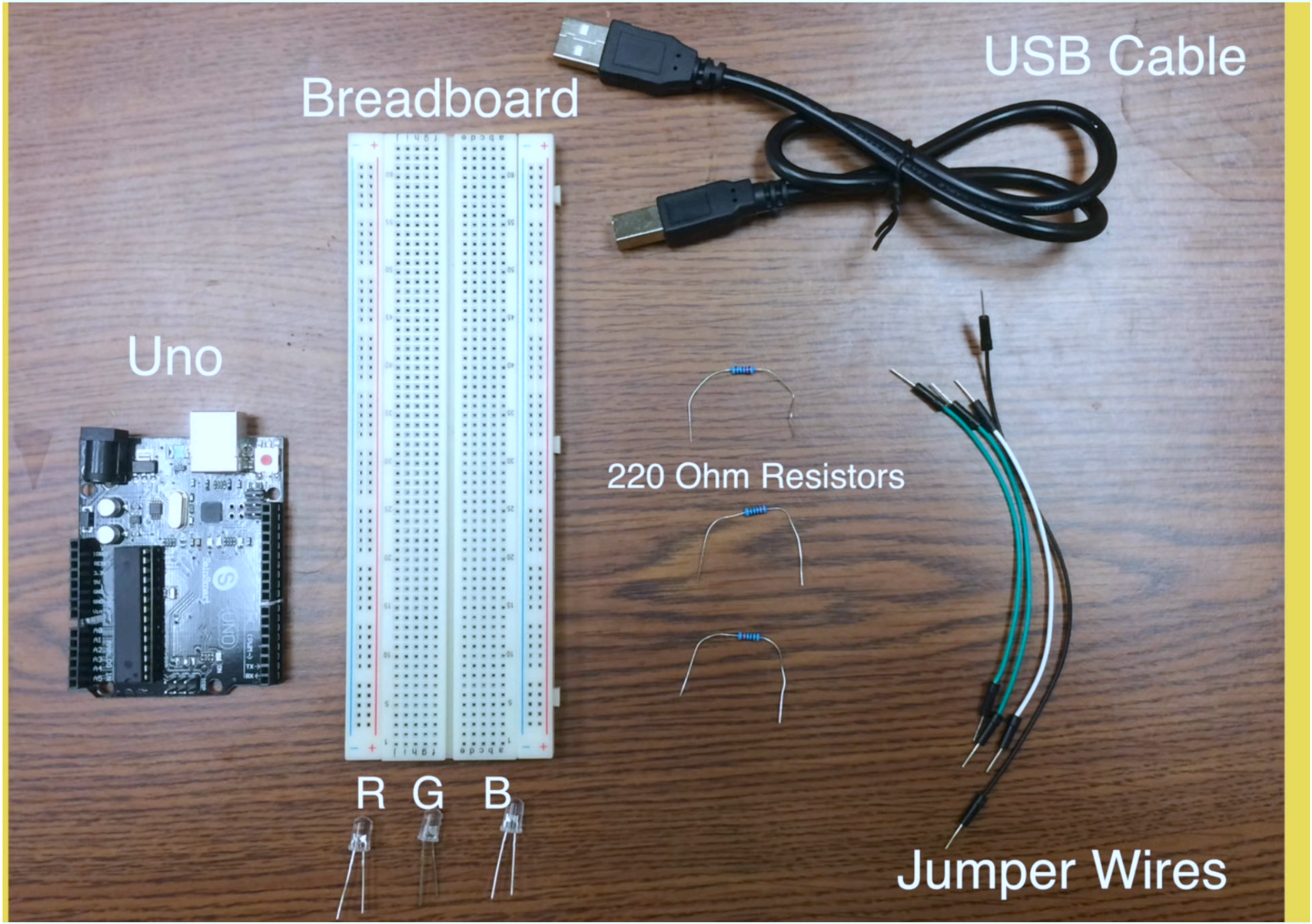
Arduino IDE installation and setup

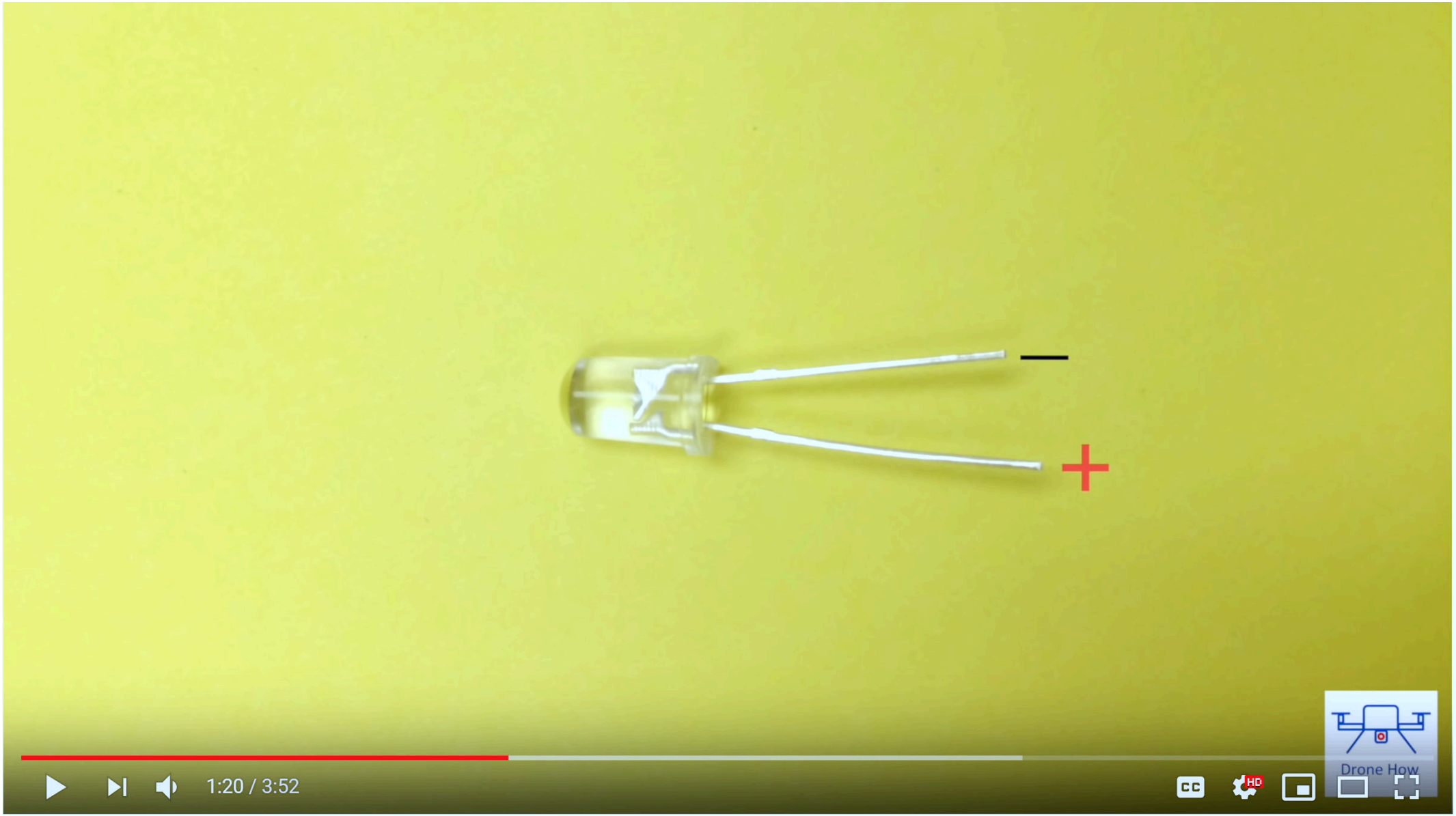
Arduino IDE is the open source Integrated Development Environment which is used for uploading programs easily to a variety of Arduino boards, clones, and compatibles.

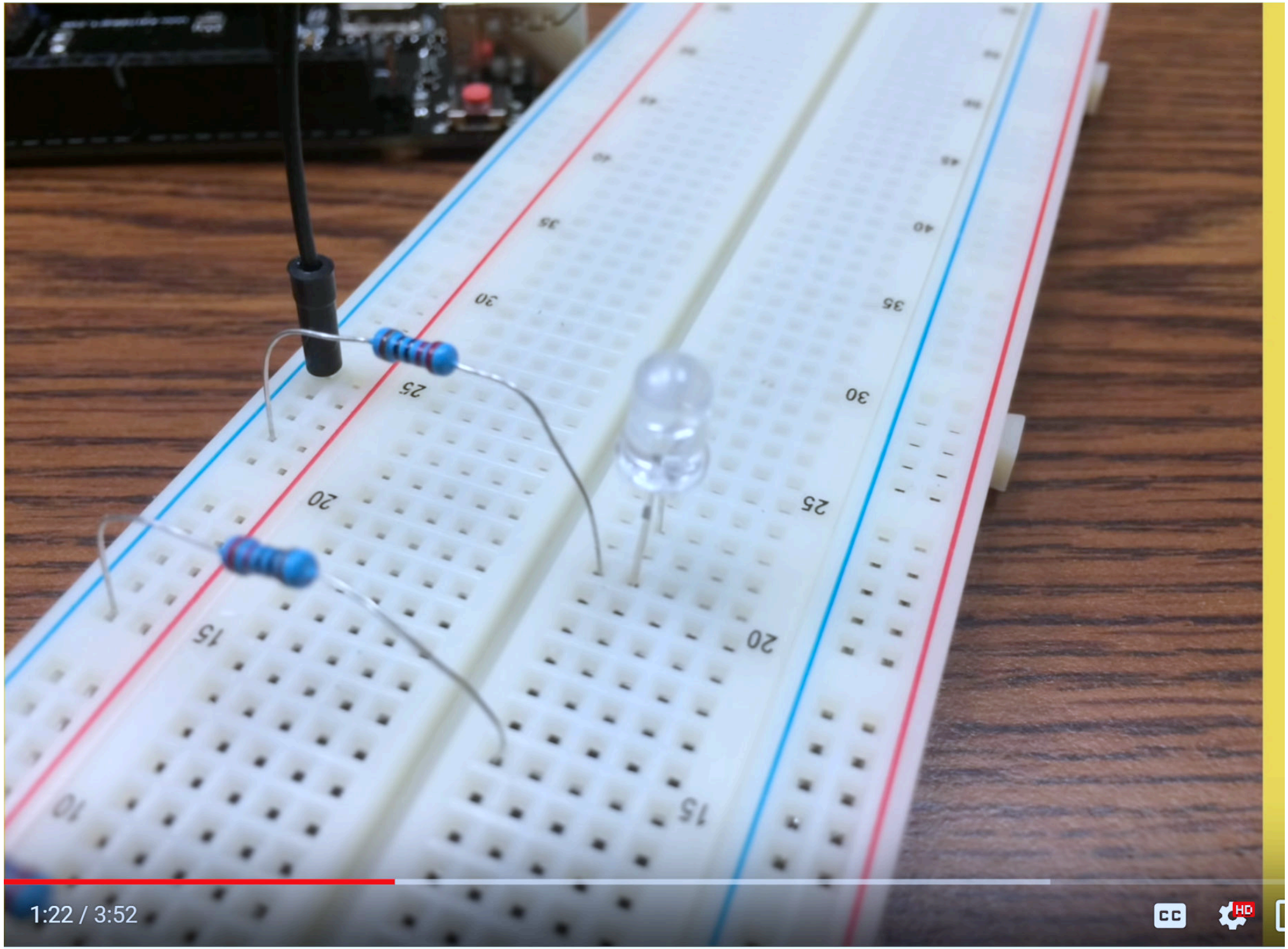
You can visit the Arduino website at https://www.arduino.cc/

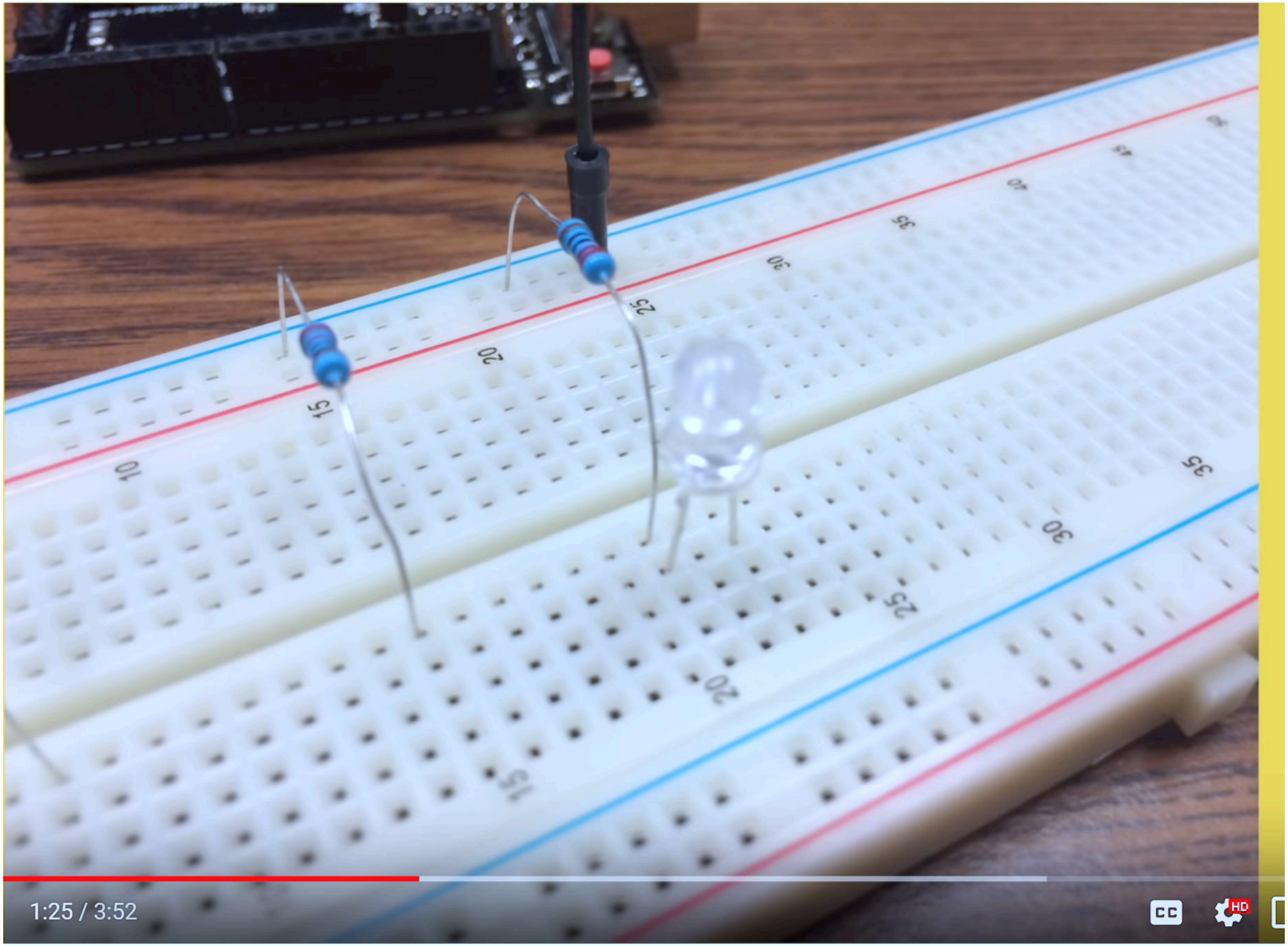


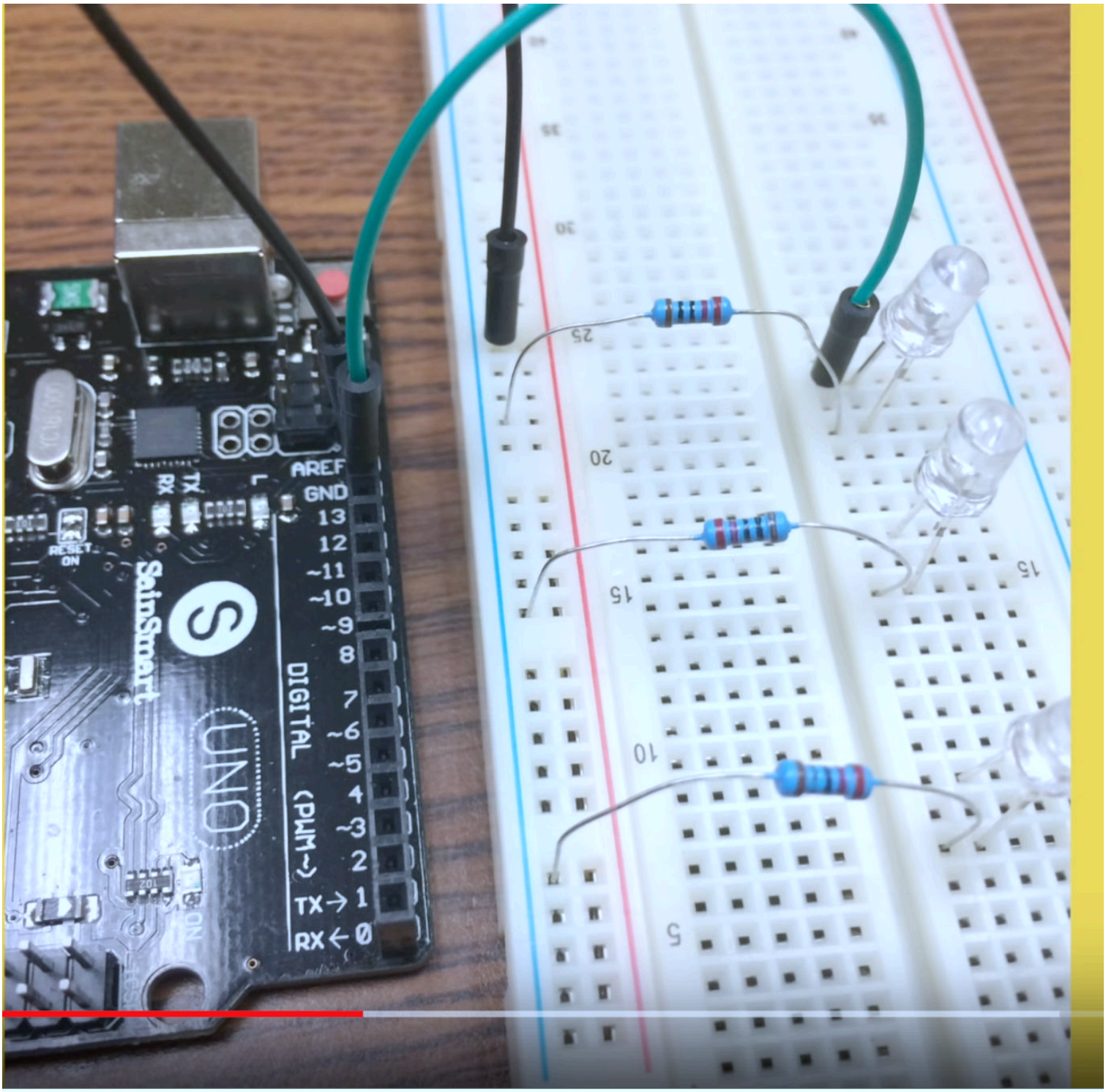
Arduino Tutorial: LED Sequential Control- Beginner Project

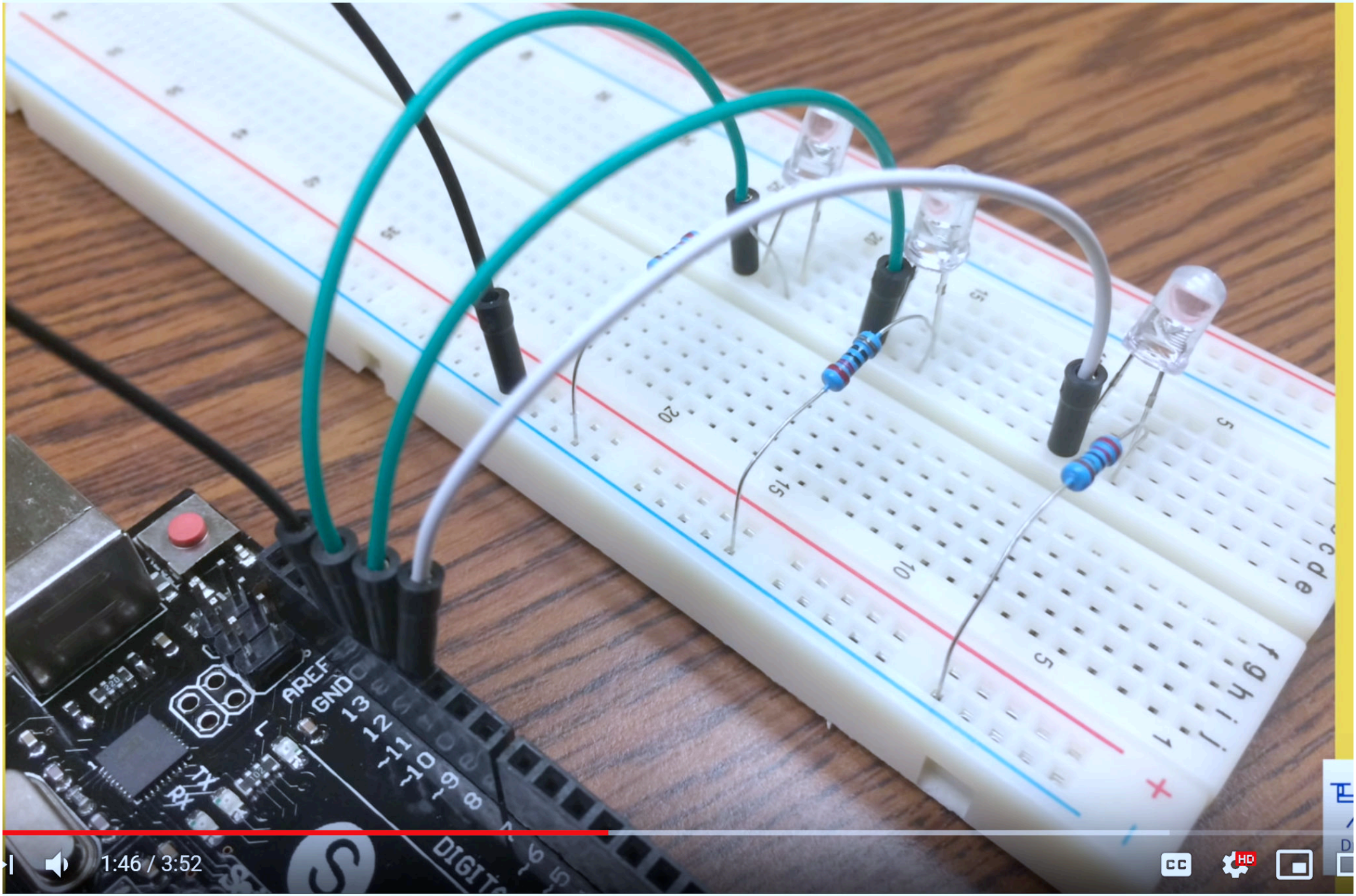

USB Cable
Breadboard
Uno
220 Ohm Resistors
R G B
Jumper Wires

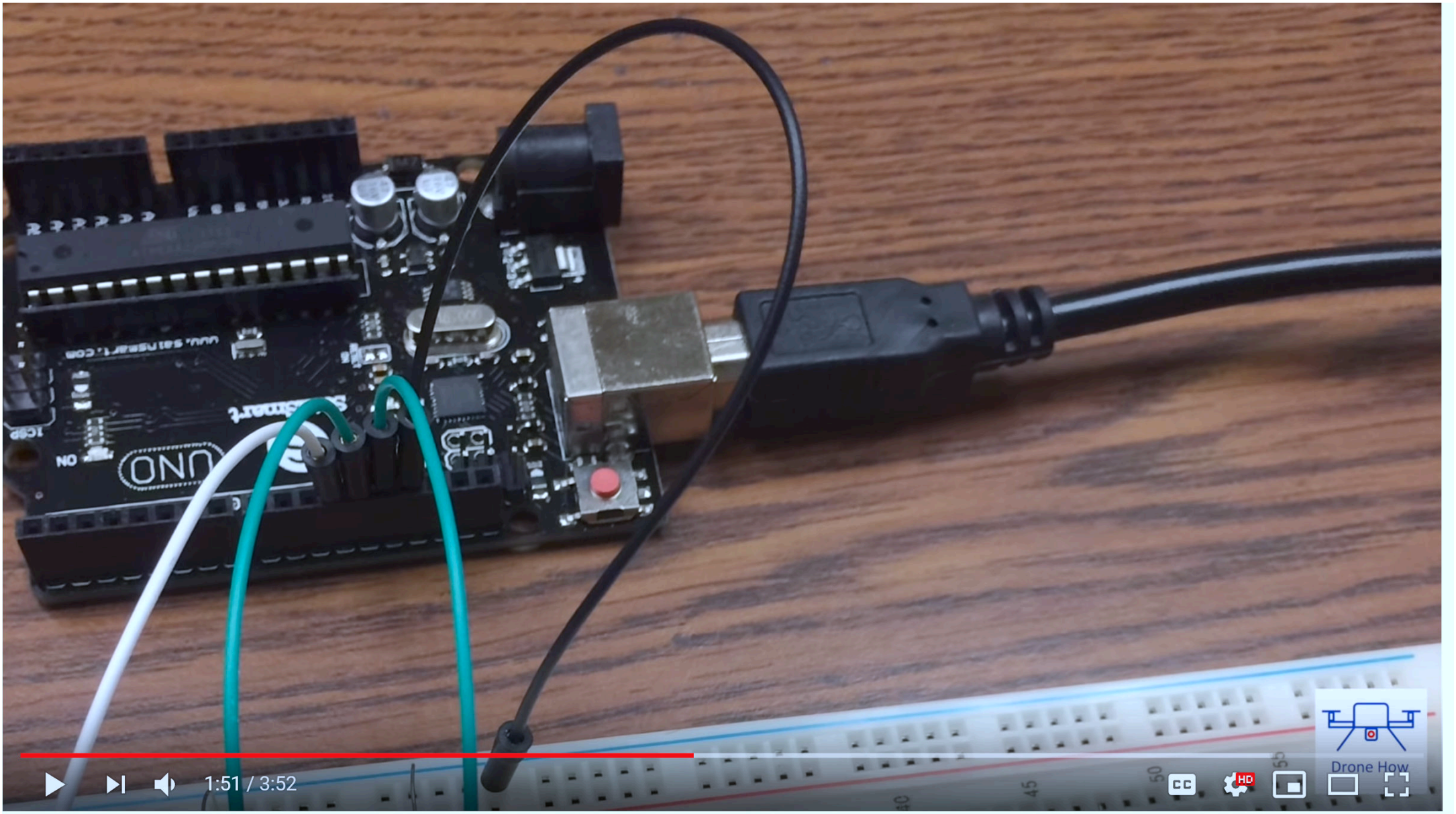

−
+
1:20 / 3:52
Drone How


1:22 / 3:52
CC
HD


1:25 / 3:52


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
SainSmart
UNO
RESET
ON
25
20
15
10
5

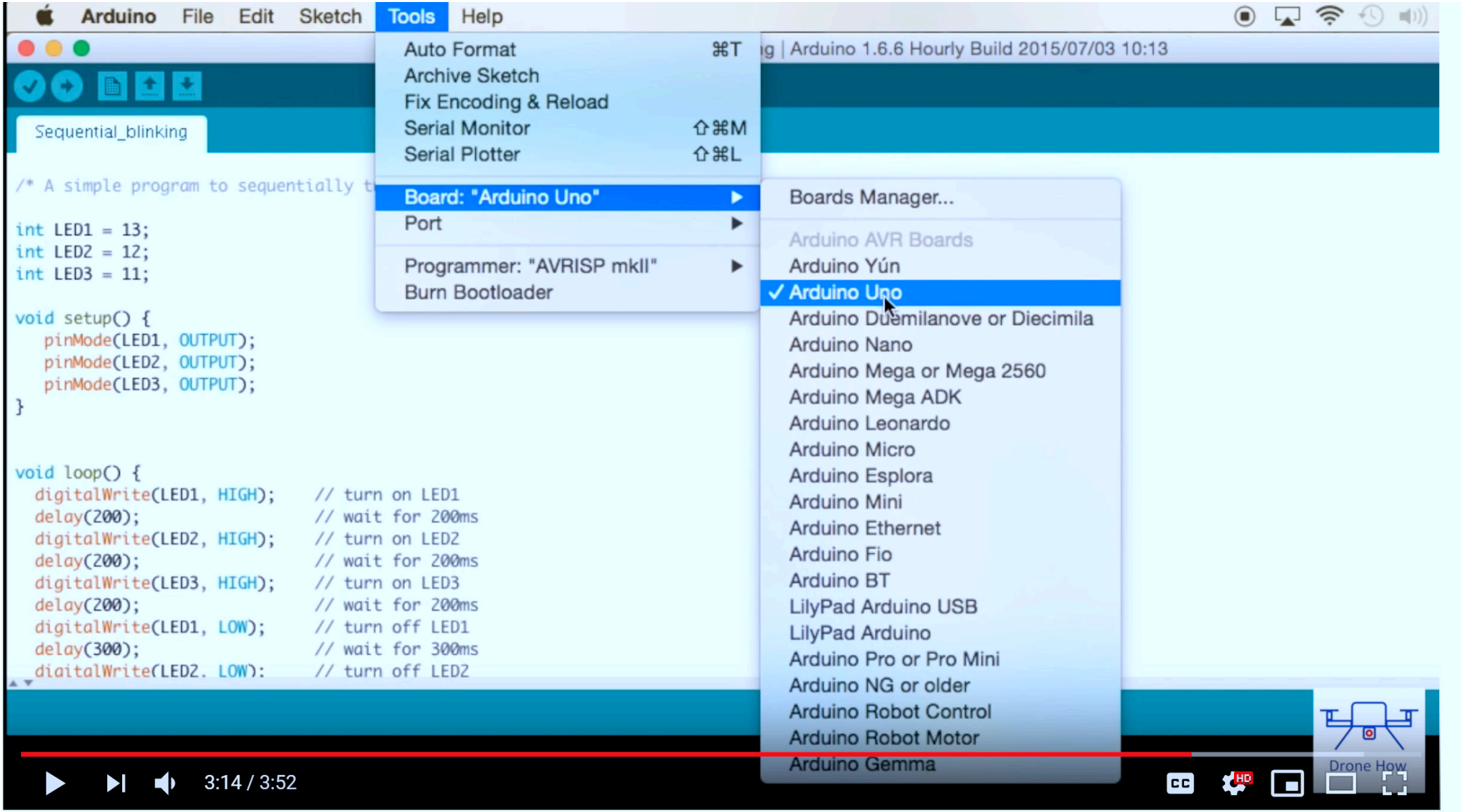

AREF
GND
13
12
~11
~10
~9
8
TX
RX
1:46 / 3:52
CC
HD


UNO
1:51 / 3:52
Drone How

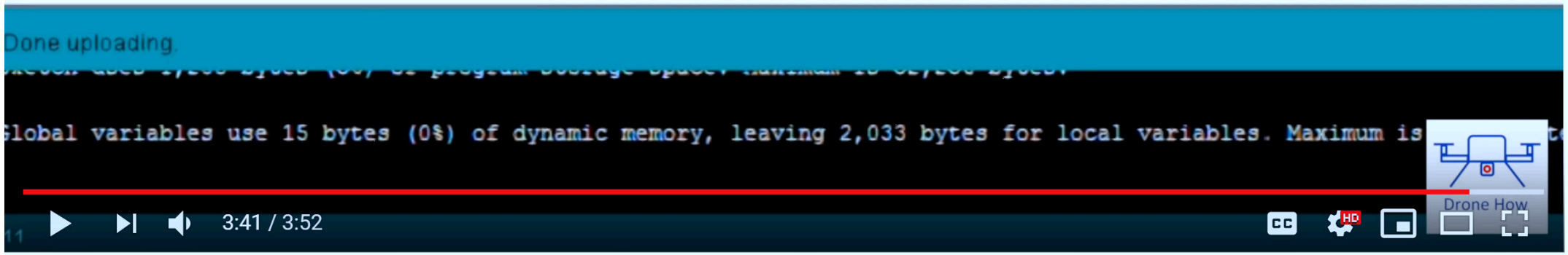
Sequential_blinking §

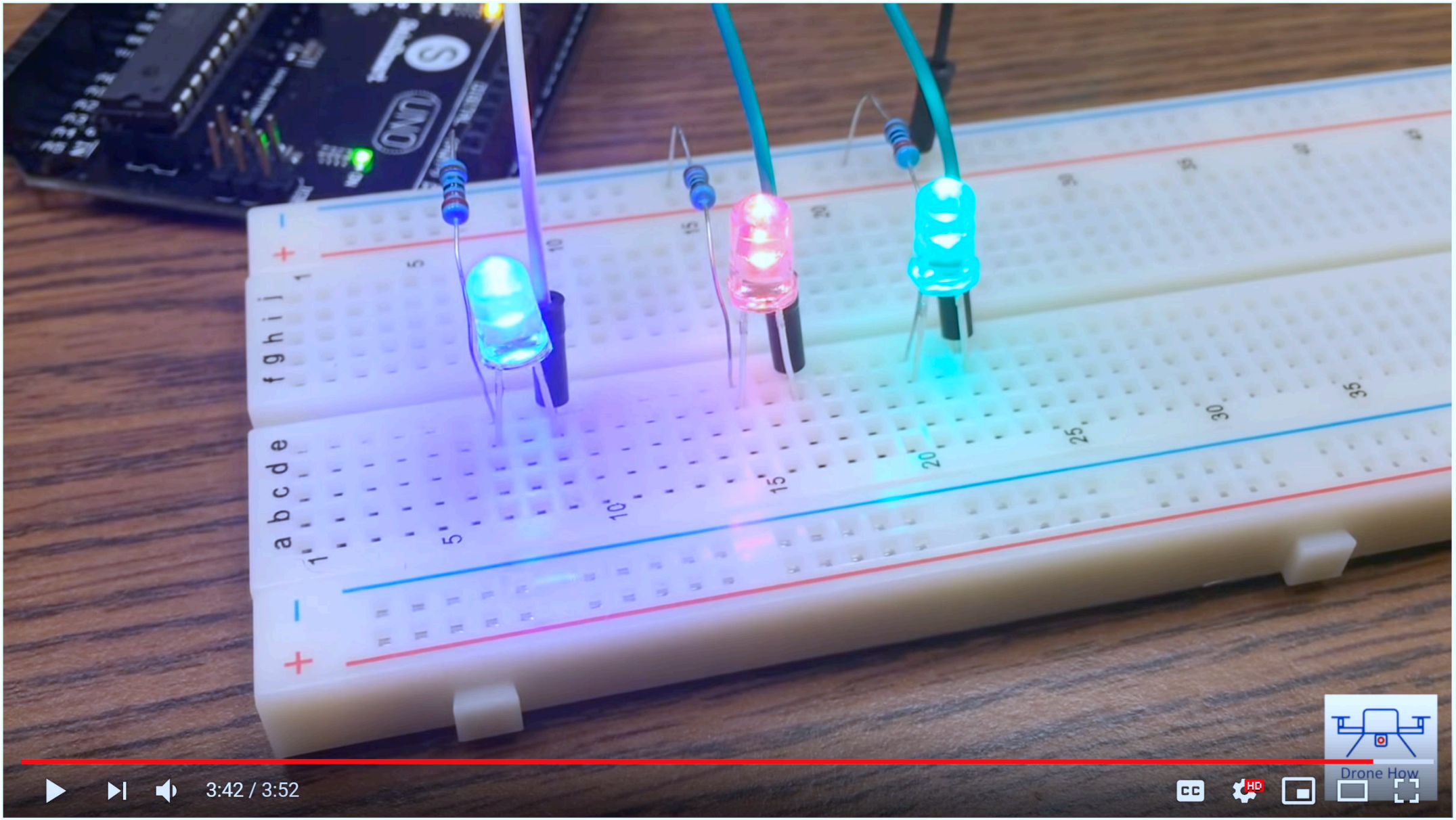
```
/* A simple program to sequentially turn on and turn off 3 LEDs */

int LED1 = 13;
int LED2 = 12;
int LED3 = 11;

void setup() {
   pinMode(LED1, OUTPUT);
   pinMode(LED2, OUTPUT);
   pinMode(LED3, OUTPUT);
}


void loop() {
  digitalWrite(LED1, HIGH);    // turn on LED1
  delay(1000);                  // wait for 200ms
  digitalWrite(LED2, HIGH);    // turn on LED2
  delay(200);                   // wait for 200ms
  digitalWrite(LED3, HIGH);    // turn on LED3
  delay(200);                   // wait for 200ms
  digitalWrite(LED1, LOW);     // turn off LED1
  delay(300);                   // wait for 300ms
  digitalWrite(LED2, LOW);     // turn off LED2
  delay(300);                   // wait for 300ms
  digitalWrite(LED3, LOW);     // turn off LED3
  delay(300);                   // wait for 300ms before running program all over again
}
```


Done uploading

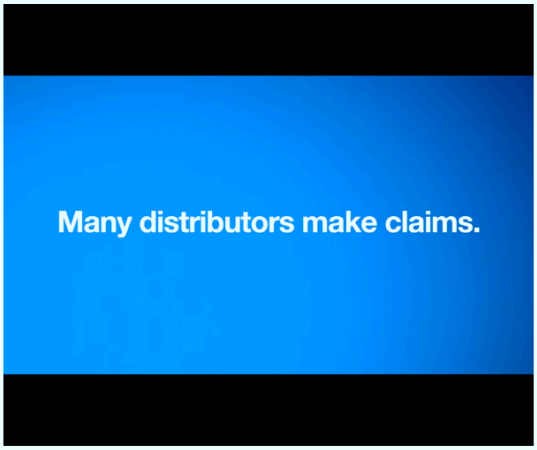

Arduino File Edit Sketch Tools Help
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Serial Plotter ⇧⌘L
Board: "Arduino Uno"
Port
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
✓ Arduino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
Arduino 1.6.6 Hourly Build 2015/07/03 10:13
Sequential_blinking
/* A simple program to sequentially t
int LED1 = 13;
int LED2 = 12;
int LED3 = 11;
void setup() {
pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(LED3, OUTPUT);
}
void loop() {
digitalWrite(LED1, HIGH); // turn on LED1
delay(200); // wait for 200ms
digitalWrite(LED2, HIGH); // turn on LED2
delay(200); // wait for 200ms
digitalWrite(LED3, HIGH); // turn on LED3
delay(200); // wait for 200ms
digitalWrite(LED1, LOW); // turn off LED1
delay(300); // wait for 300ms
digitalWrite(LED2, LOW); // turn off LED2
3:14 / 3:52
Drone How

```
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
}


void loop() {
  digitalWrite(LED1, HIGH);     // turn on LED1
  delay(200);                   // wait for 200ms
  digitalWrite(LED2, HIGH);     // turn on LED2
  delay(200);                   // wait for 200ms
  digitalWrite(LED3, HIGH);     // turn on LED3
  delay(200);                   // wait for 200ms
  digitalWrite(LED1, LOW);      // turn off LED1
  delay(300);                   // wait for 300ms
  digitalWrite(LED2, LOW);      // turn off LED2
  delay(300);                   // wait for 300ms
  digitalWrite(LED3, LOW);      // turn off LED3
  delay(300);                   // wait for 300ms before running program all over again
}
```




UNO
a b c d e
f g h i j
3:42 / 3:52
Drone How

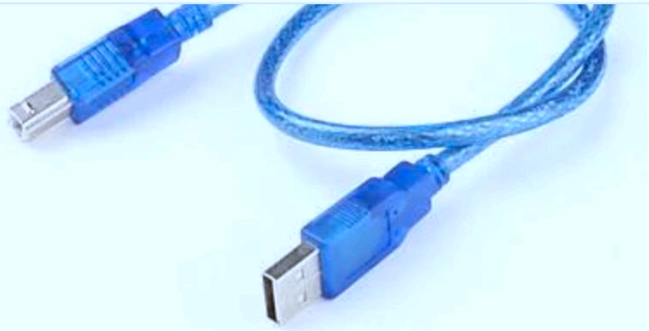
Interface a Raspberry Pi with an Arduino so the two boards can communicate with one another.



ARTICLE

Sometimes you may need to connect an Arduino to a Raspberry Pi. For example, if you have sensors, motors, and actuators, you can connect these to the Arduino and make the Arduino send values to and from the Raspberry Pi. This way, we can separate the computing intensive tasks (done by the Raspberry Pi) and controlling tasks (done by the Arduino).

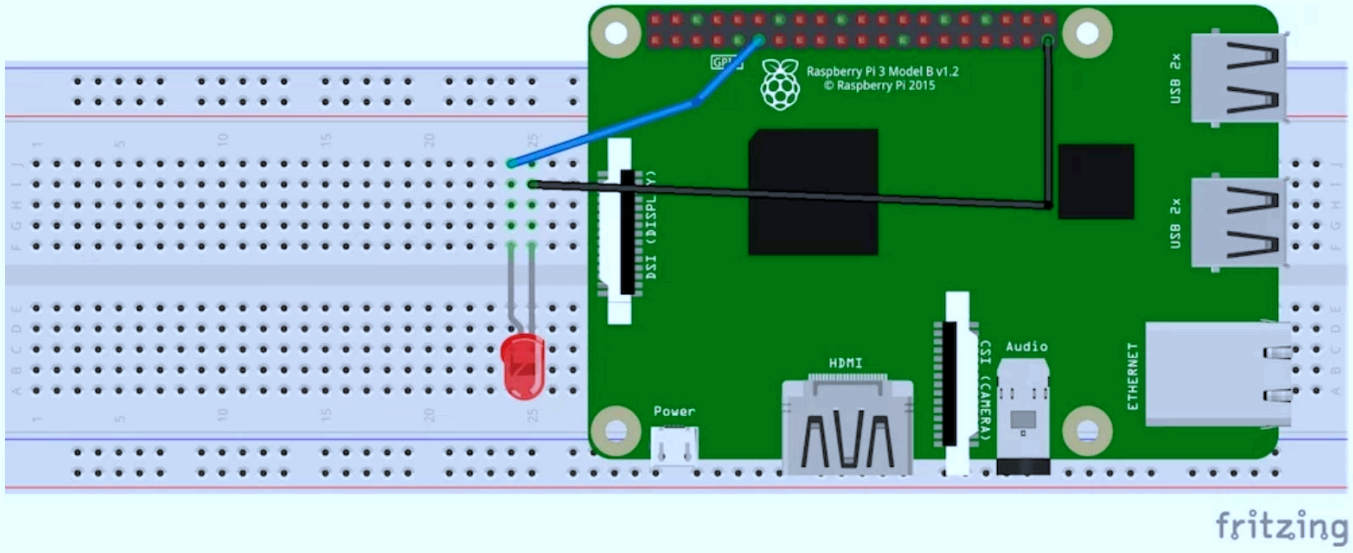
In this tutorial, we will connect an Arduino to a Raspberry Pi and have the Arduino send “Hello from Arduino” to the Raspberry Pi, and the Raspberry Pi will blink an LED upon receiving the command from the Arduino.



For communication, we will use simple serial communication over USB cable.

So, let's get started!

Connect the LED to pin number 11 as shown in the picture below.



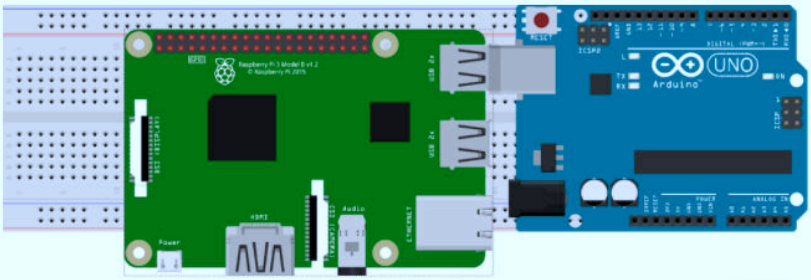
Turn on the Raspberry Pi and open Python 3 in a new window.

Write the following code in the new window and save it. (Save to your desktop so you don't lose it.)

```python
import serial
import RPi.GPIO as GPIO
import time

ser=serial.Serial("/dev/ttyACM0",9600)  #change ACM number as found from ls /
ser.baudrate=9600
def blink(pin):


GPIO.output(pin,GPIO.HIGH)
```

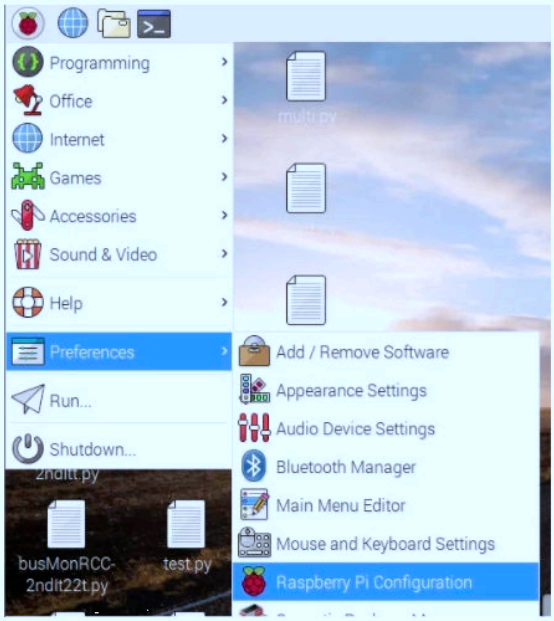
Now open Arduino IDE and upload the following code to your Arduino.

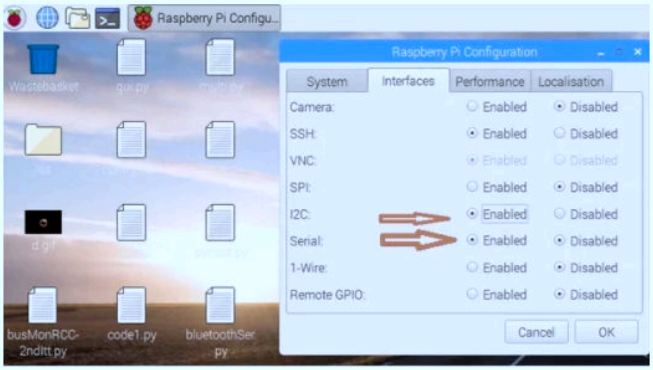
```arduino
String data="Hello From Arduino!";

void setup() {
// put your setup code here, to run once:
Serial.begin(9600);

}

void loop() {
// put your main code here, to run repeatedly:
```



Make sure the code is uploaded to Arduino.

In your Raspberry Pi interface, be sure to enable Serial and I2C in PiConfig.





Next, you'll need to restart your Raspberry Pi. Open the Terminal and execute these commands:

```bash
sudo apt-get install python-serial

sudo pip install pyserial
```

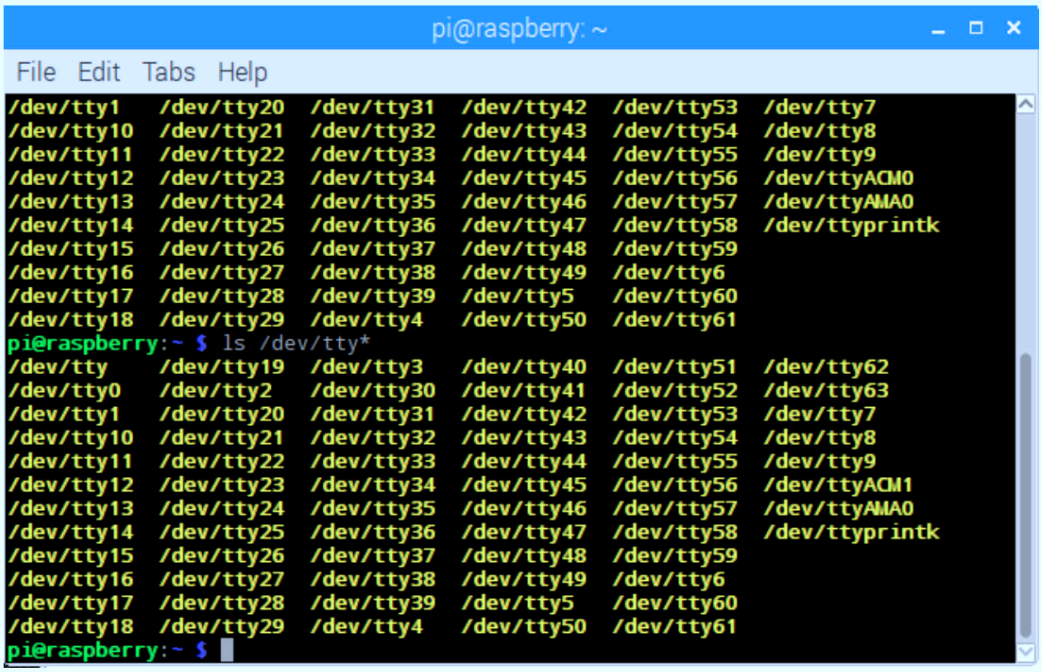
Connect your Arduino to your Raspberry Pi.

Execute.

```bash
ls /dev/tty*
```

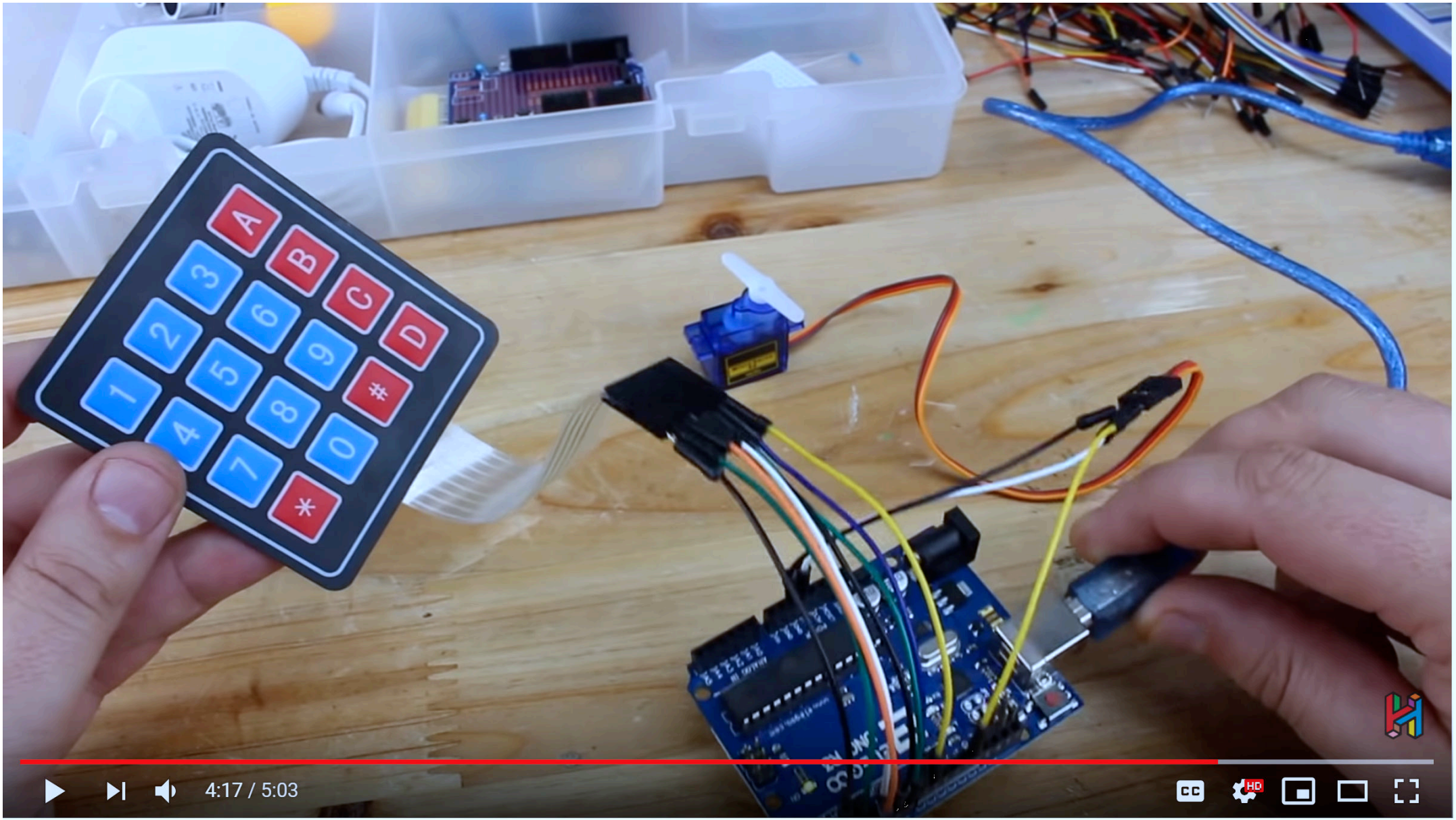
Copy

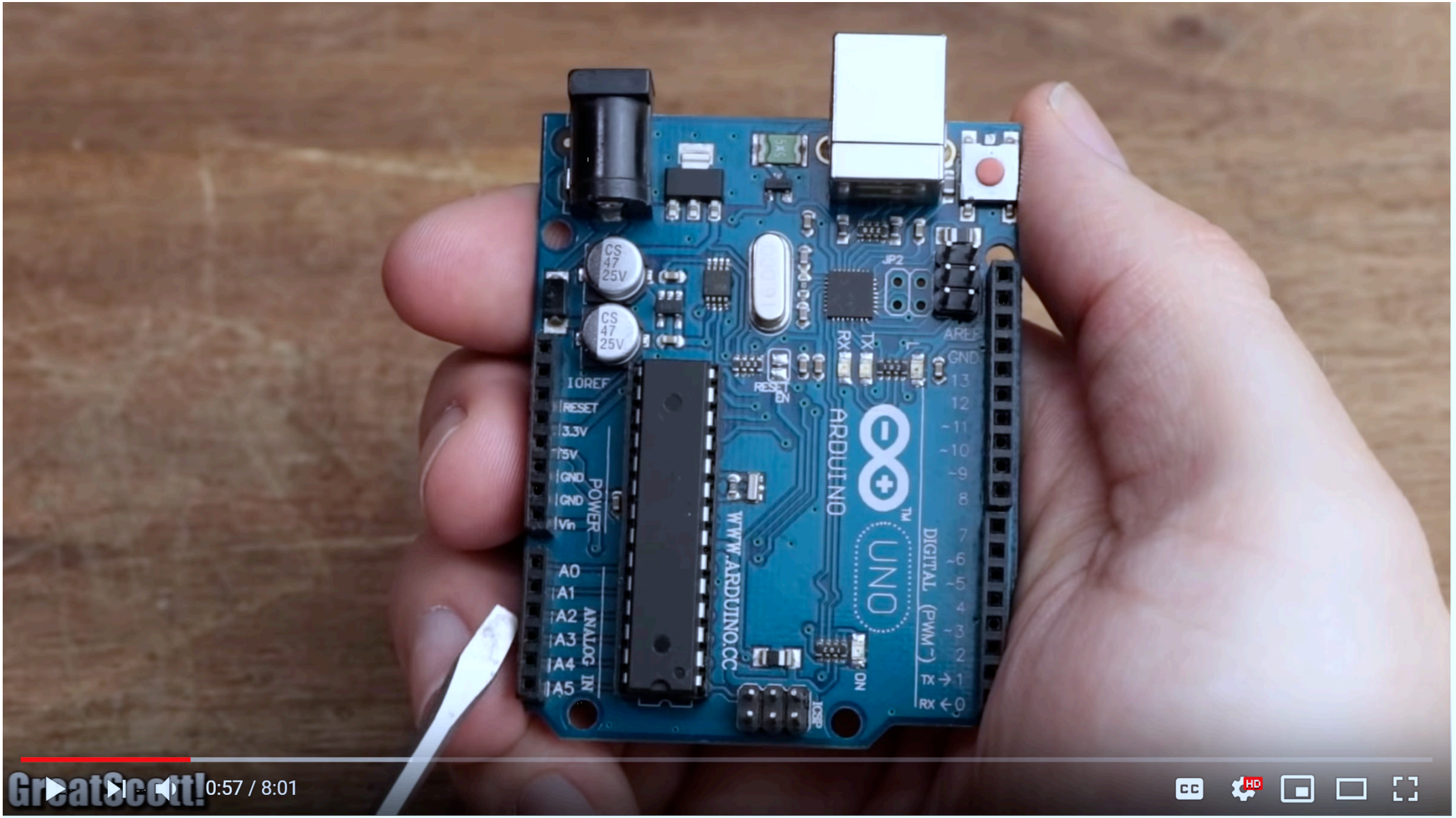
Then find a line with **/dev/ttyACM0** or something like **/dev/ttyACM1** etc. (check for an ACM with any number 0,1,2 etc.)

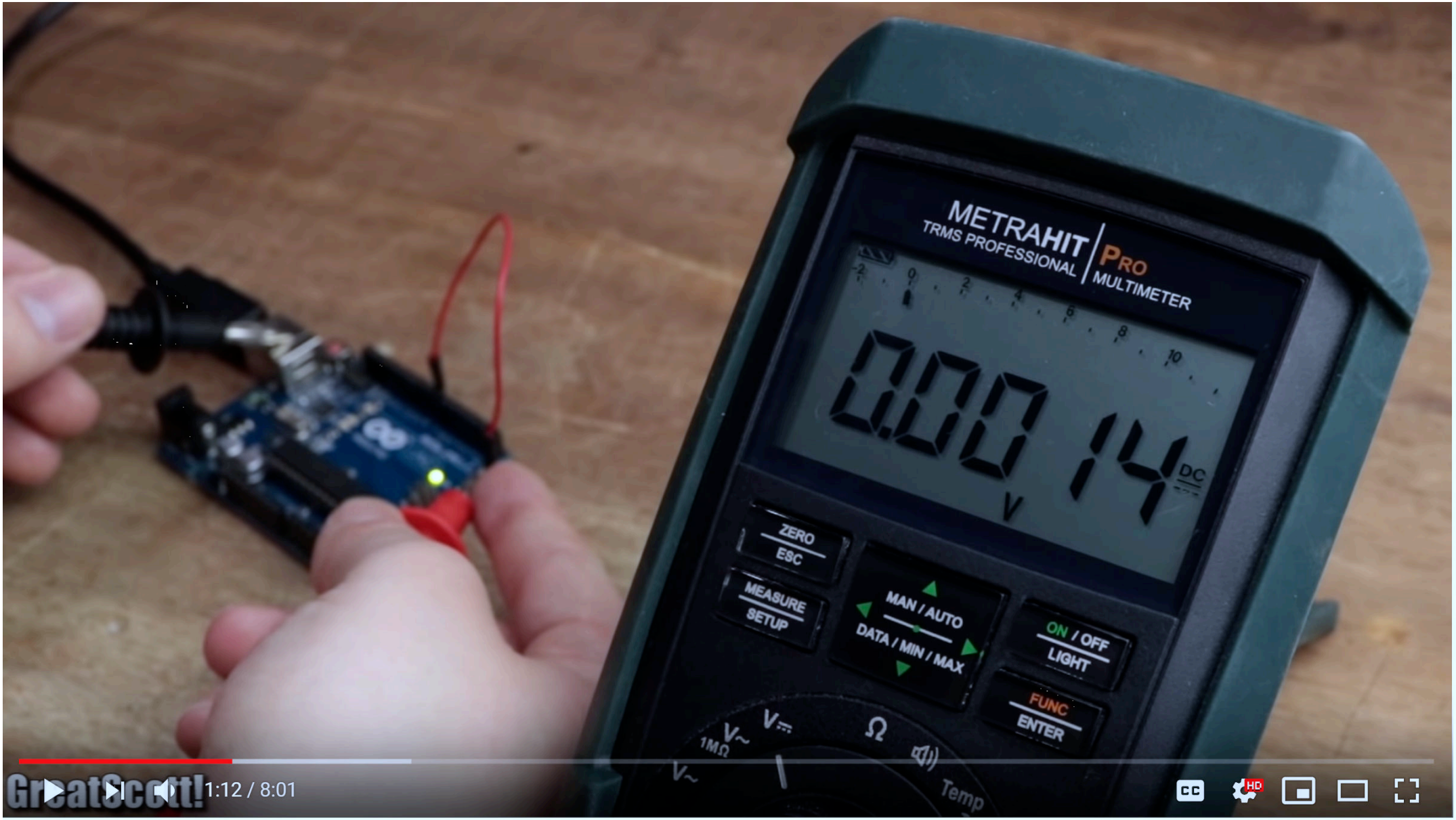


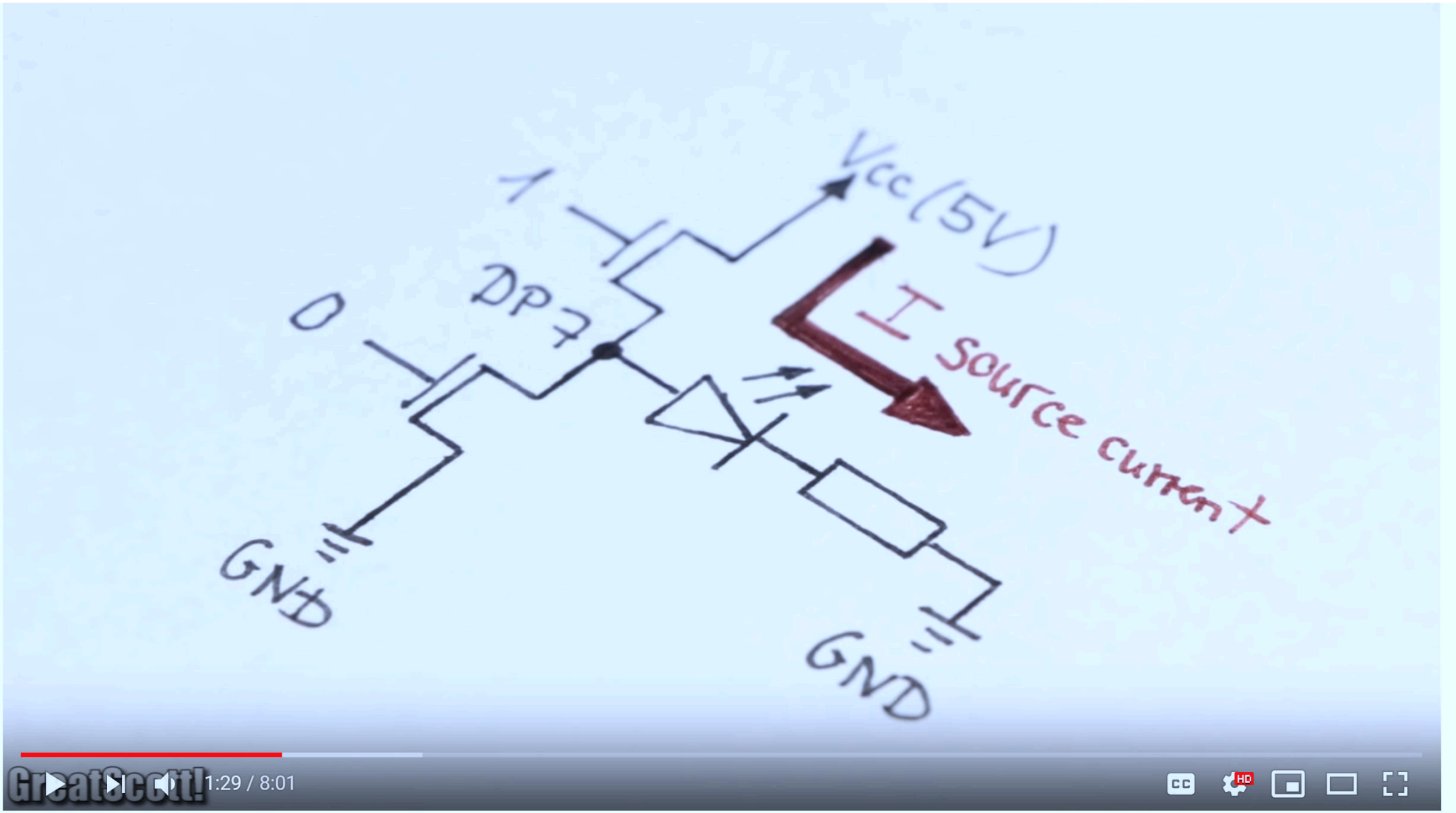
Open Python again and change **ser=serial.Serial("dev/ttyACM1",9600)** to the ACM number you found. So, if in your case you got **ACM0**, the line should look like this: **ser=serial.Serial("dev/ttyACM0",9600)**

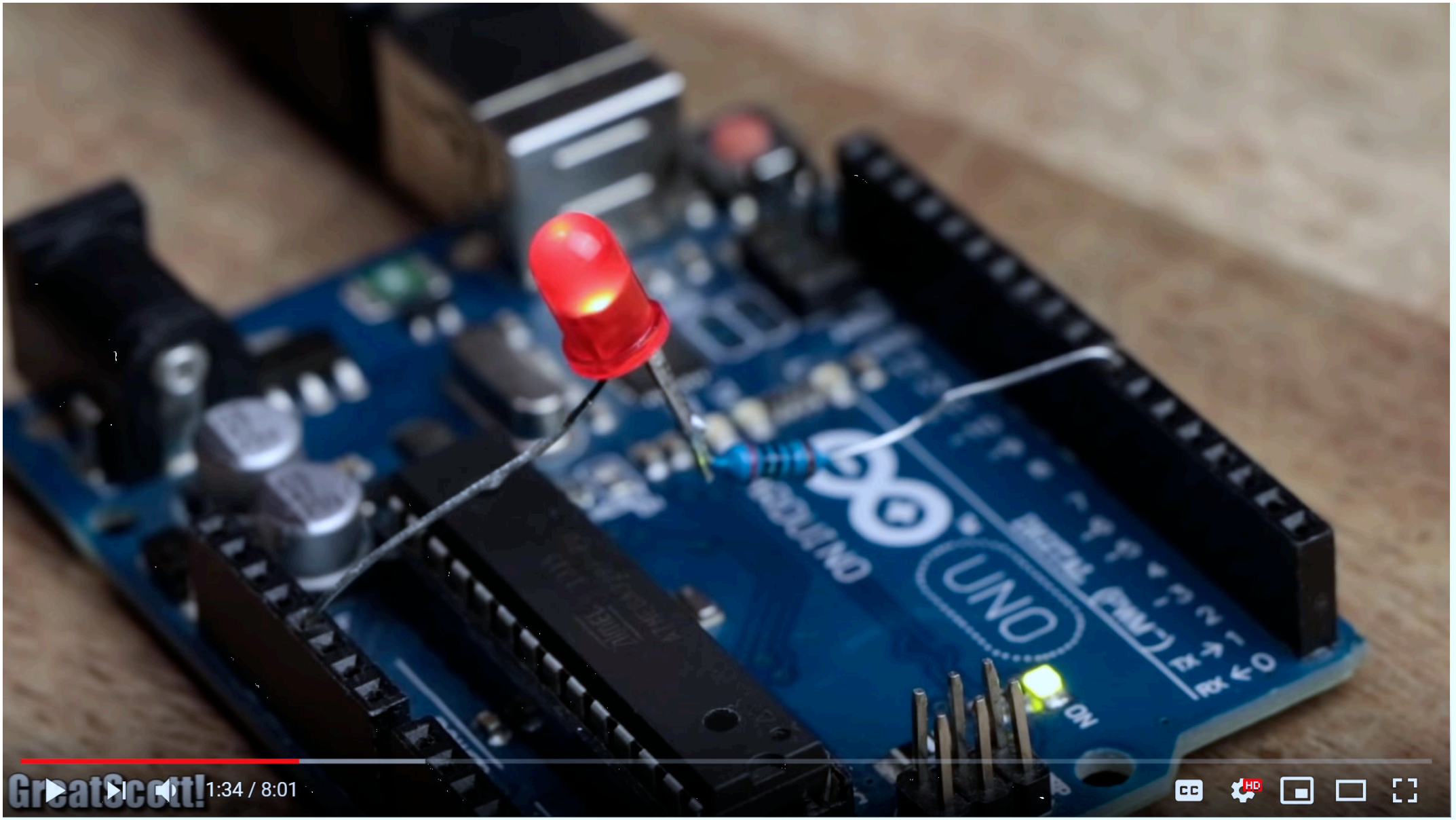
Now run the program you just created in Python3. You will see "Hello From Arduino!" in the Python terminal, and your LED should be blinking as well!

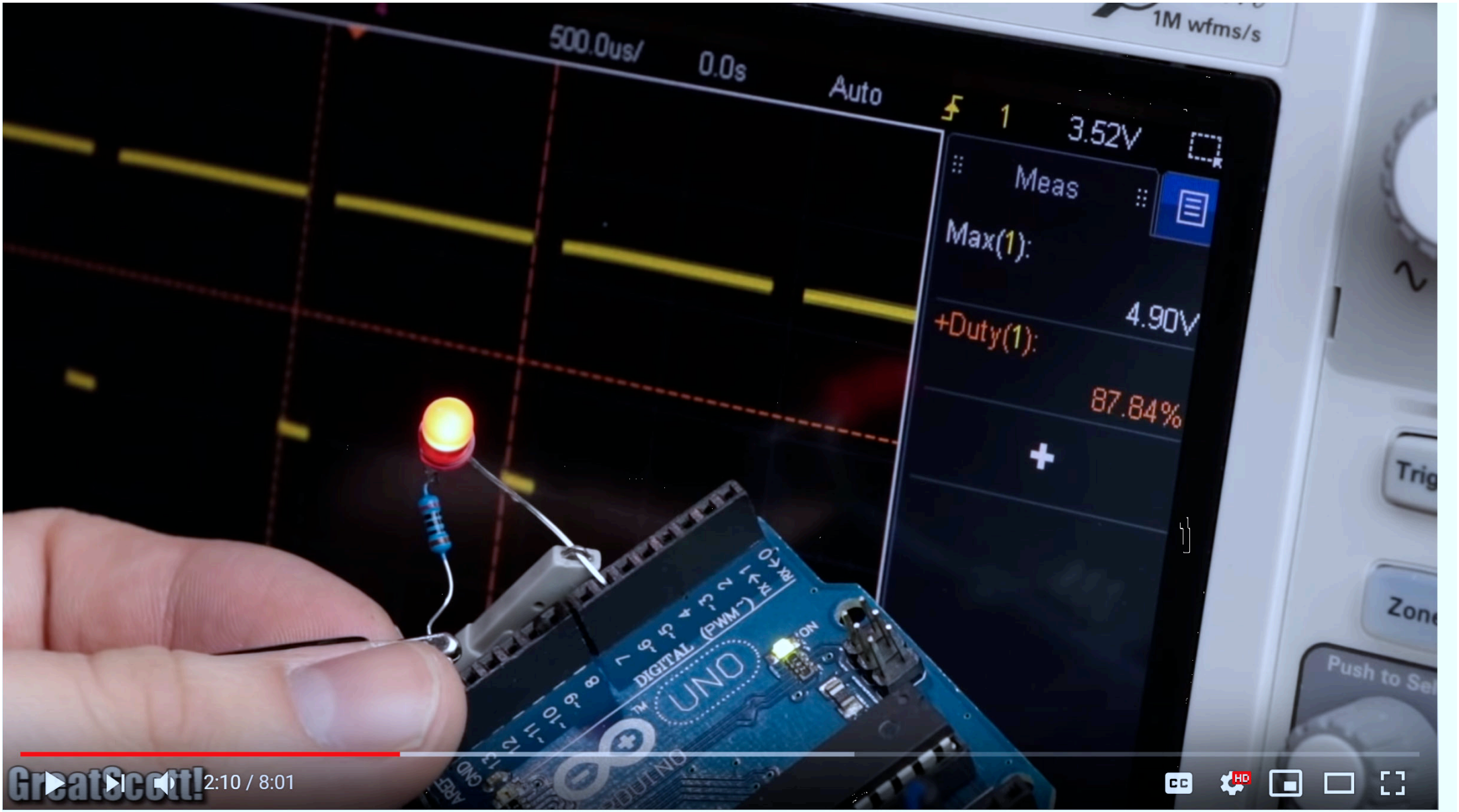

A
B
C
D
3
6
9
#
2
5
8
0
1
4
7
*
4:17 / 5:03
CC
HD

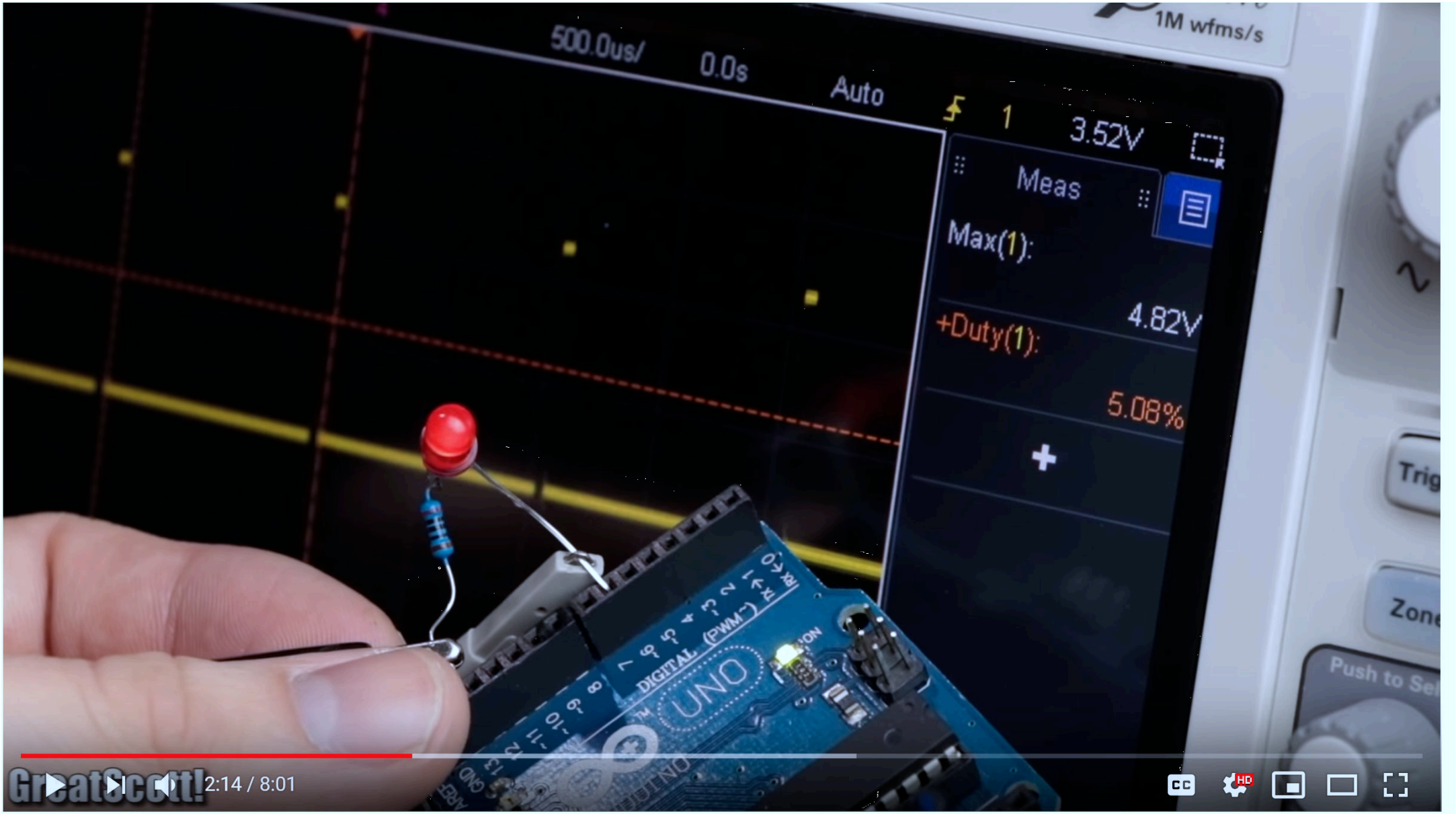

GreatScott!
0:57 / 8:01
CS 47 25V
CS 47 25V
JP2
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
RESET
TX
RX
L
ON
ARDUINO
UNO
WWW.ARDUINO.CC
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
HD

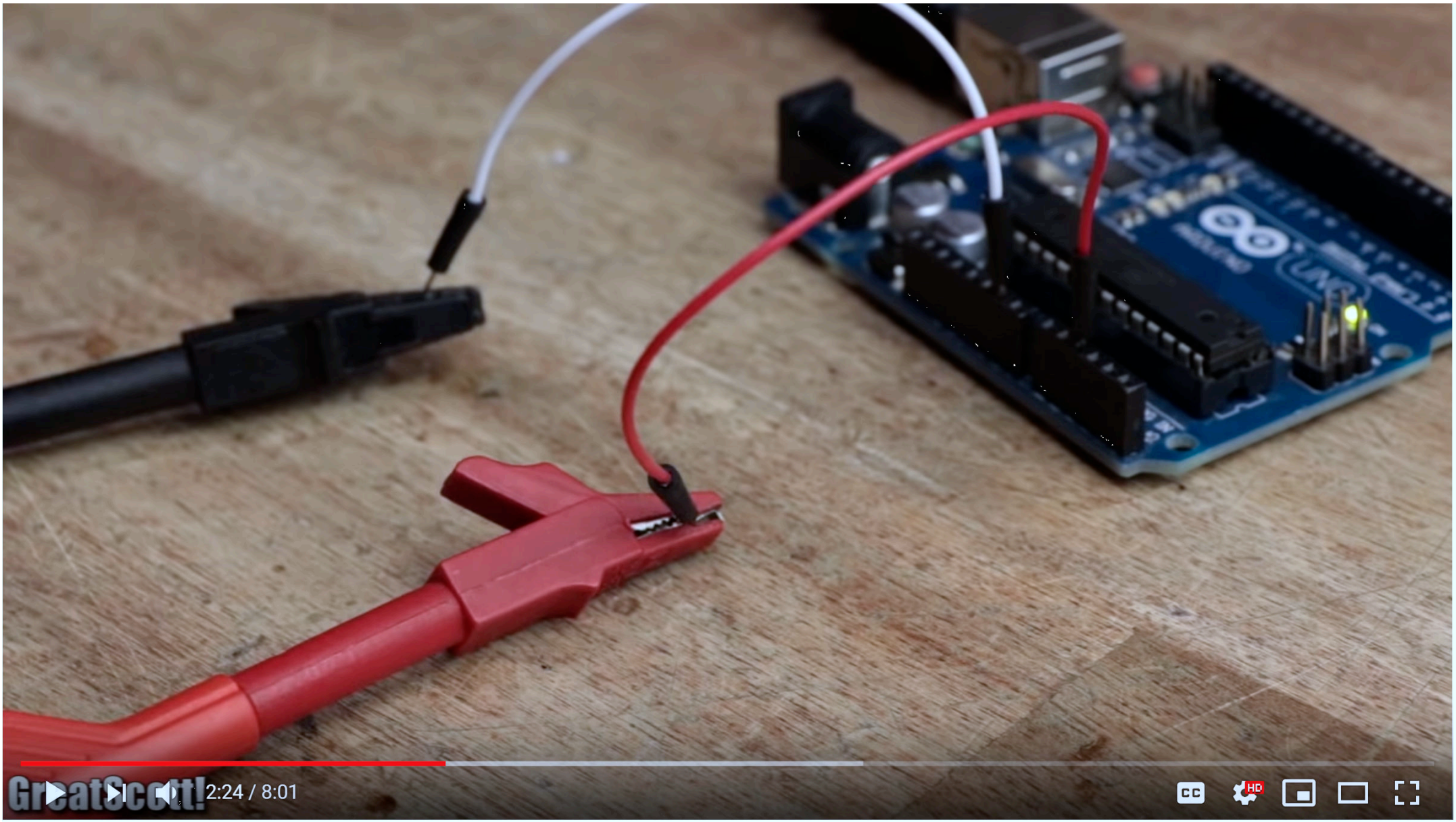

METRAHIT
TRMS PROFESSIONAL
PRO
MULTIMETER
0.00 14
V
DC
ZERO
ESC
MEASURE
SETUP
MAN / AUTO
DATA / MIN / MAX
ON / OFF
LIGHT
FUNC
ENTER
1MΩ
Temp
GreatScott!
1:12 / 8:01

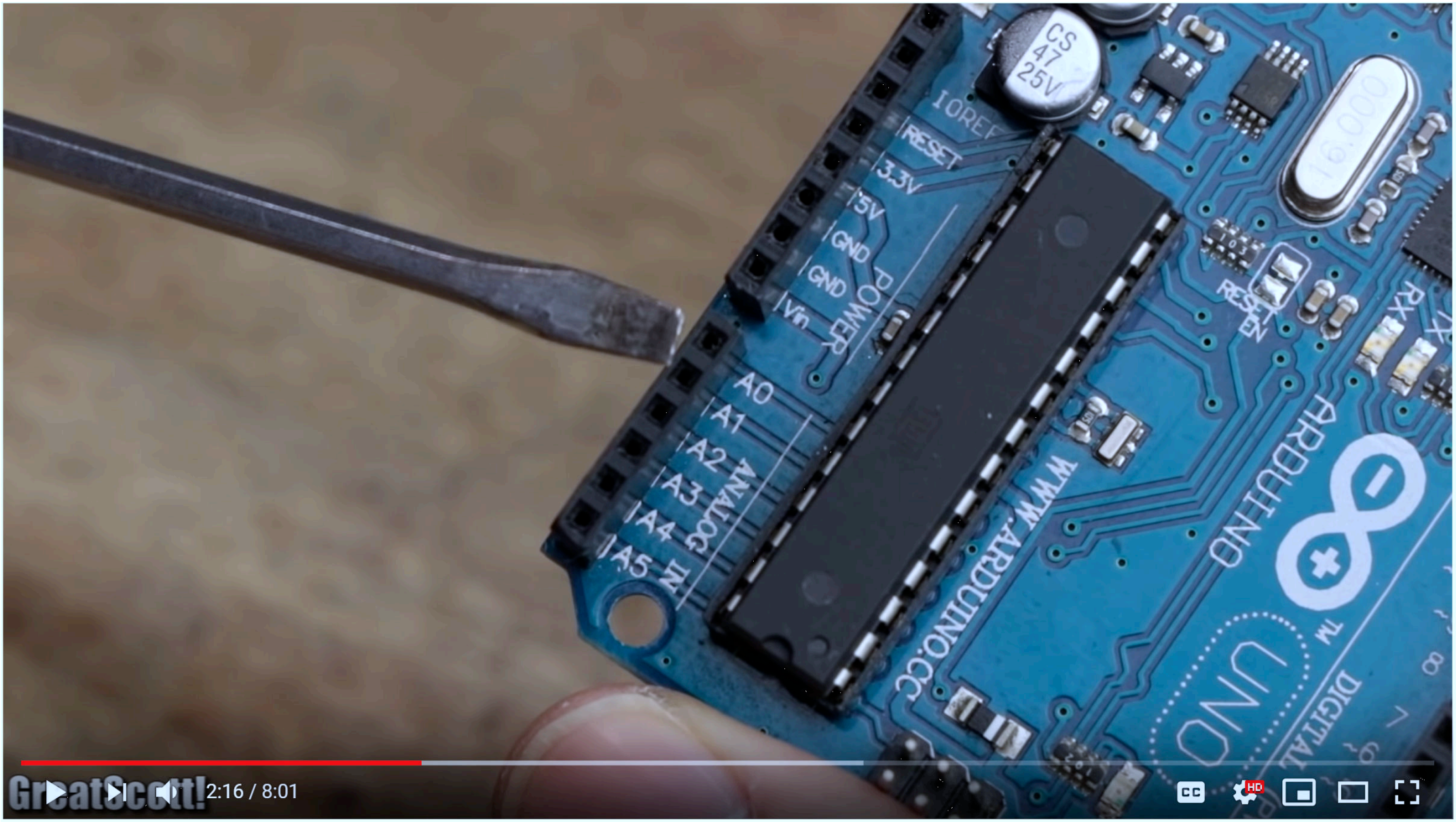

1
0
DP7
Vcc (5V)
I source current
GND
GND
GreatScott!
1:29 / 8:01


UNO
ARDUINO
DIGITAL (PWM~)
TX → 1
RX ← 0
ON
GreatScott!
1:34 / 8:01

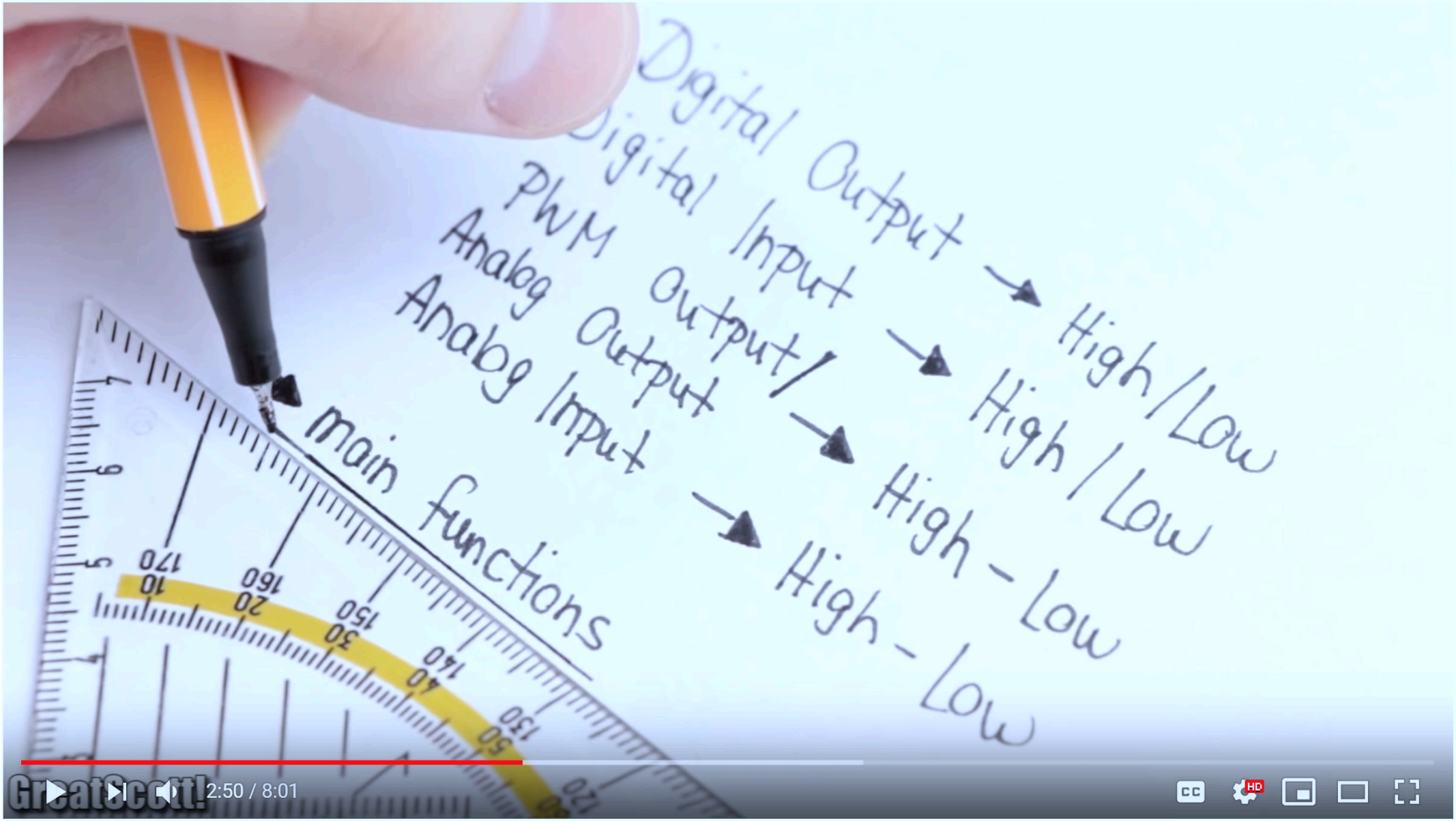

500.0us/
0.0s
Auto
1
3.52V
1M wfms/s
Meas
Max(1):
4.90V
+Duty(1):
87.84%
Trig
Zone
Push to Se
DIGITAL (PWM~)
UNO
ON
GreatScott!
2:10 / 8:01

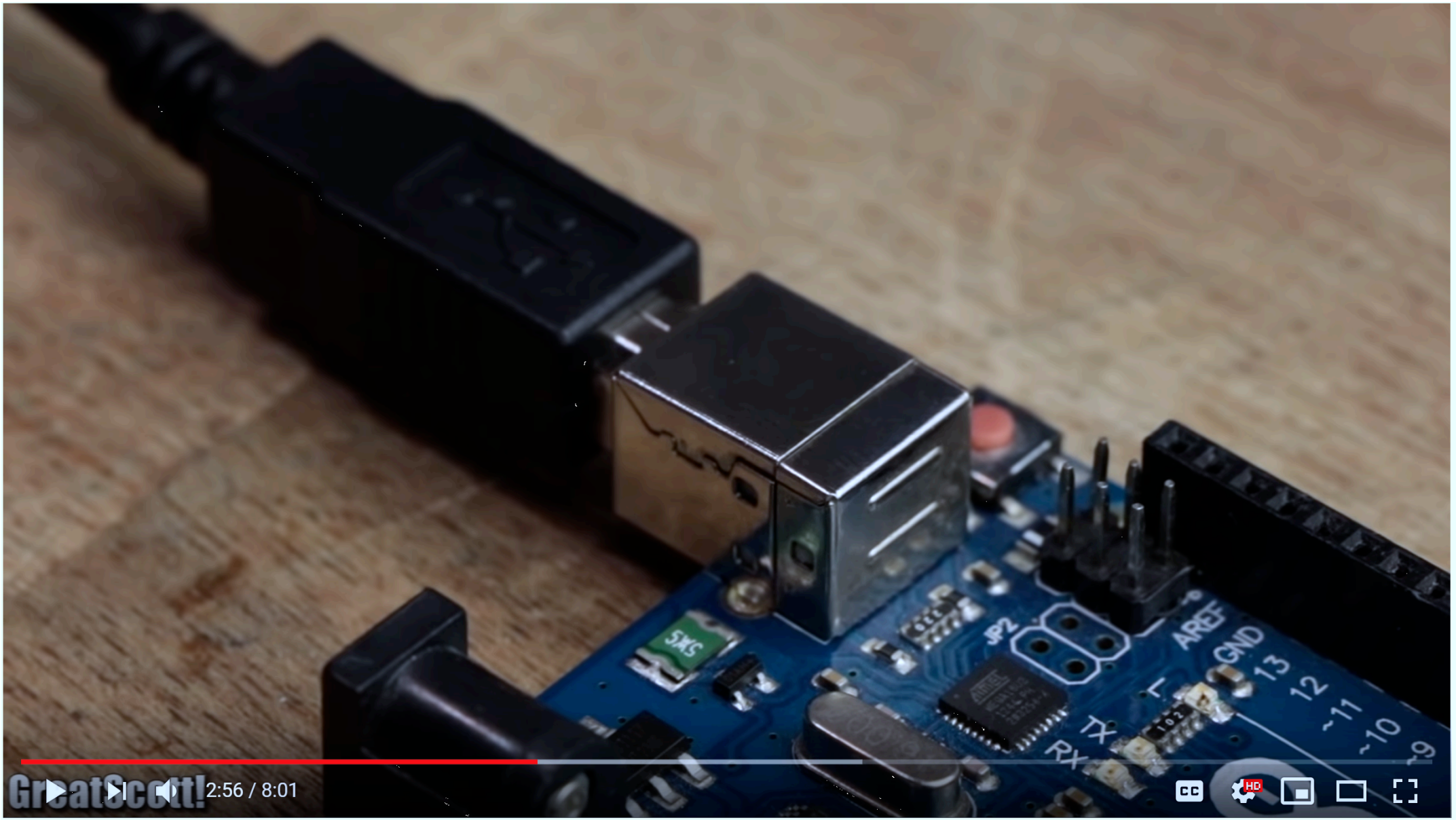

500.0us/
0.0s
Auto
1
3.52V
1M wfms/s
Meas
Max(1):
4.82V
+Duty(1):
5.08%
Trig
Zone
Push to Sel
DIGITAL (PWM~)
UNO
GreatScott!
2:14 / 8:01

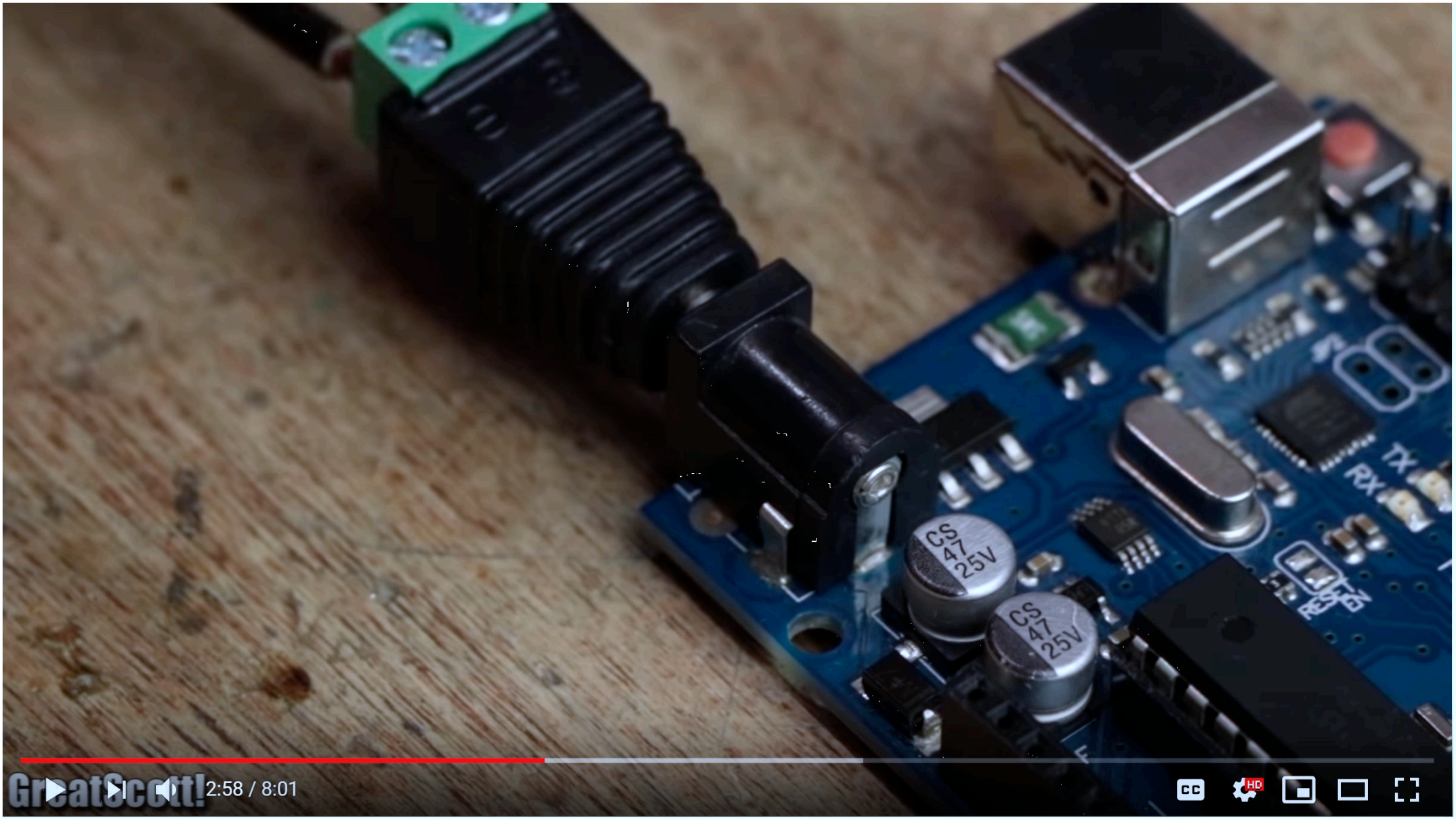

GreatScott!
2:24 / 8:01

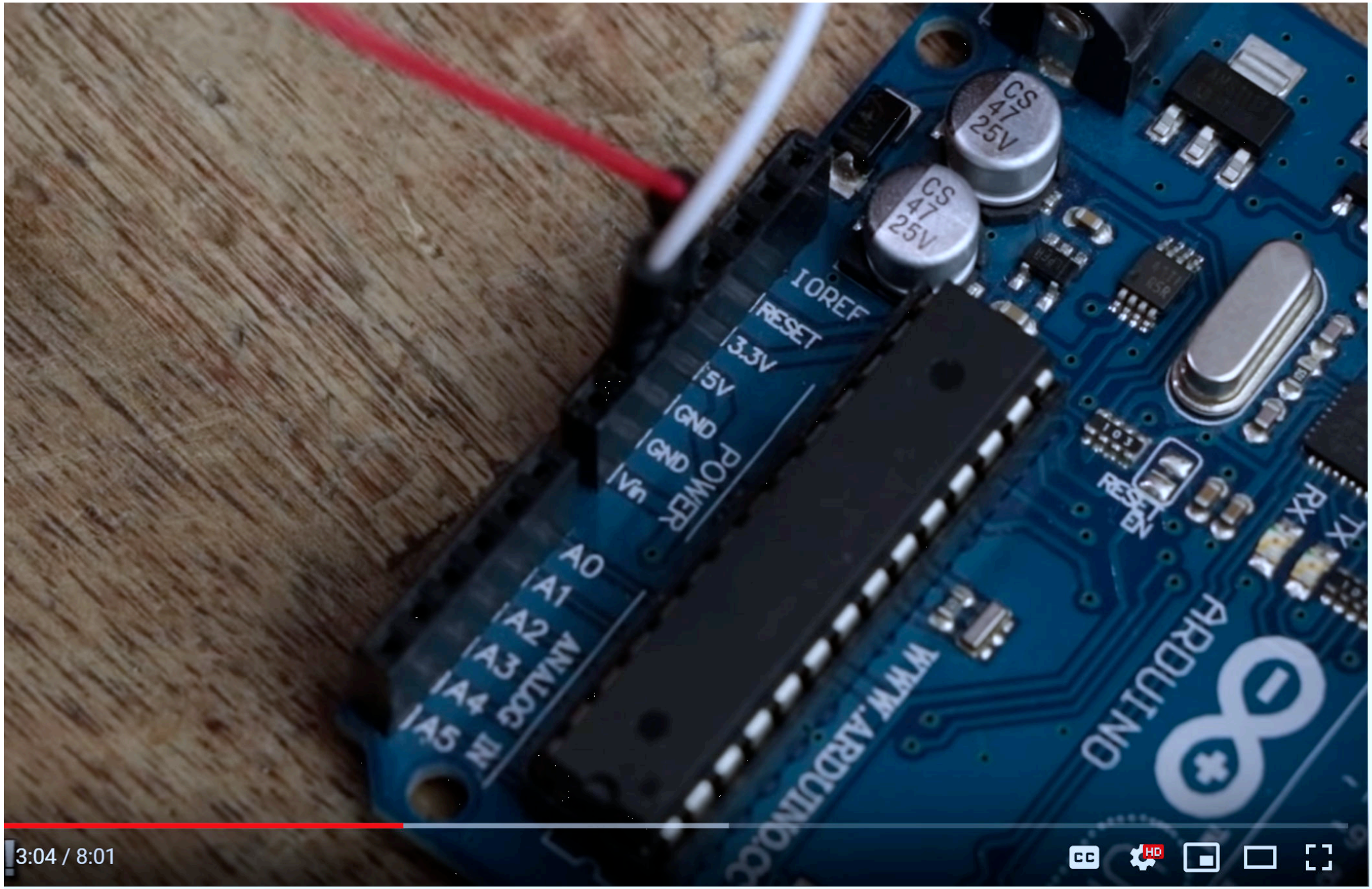

CS
47
25V
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
RESET
EN
RX
TX
WWW.ARDUINO.CC
ARDUINO
UNO
DIGITAL
GreatScott!
2:16 / 8:01


ELV
DPS 5315
Master
0.90 V
Active
U-Limit
0.00 V
0.000 A
I-Limit
0.000 A
0.00 W
Master
Slave
Dual
Series
U / I
Recall
GreatScott!
2:26 / 8:01

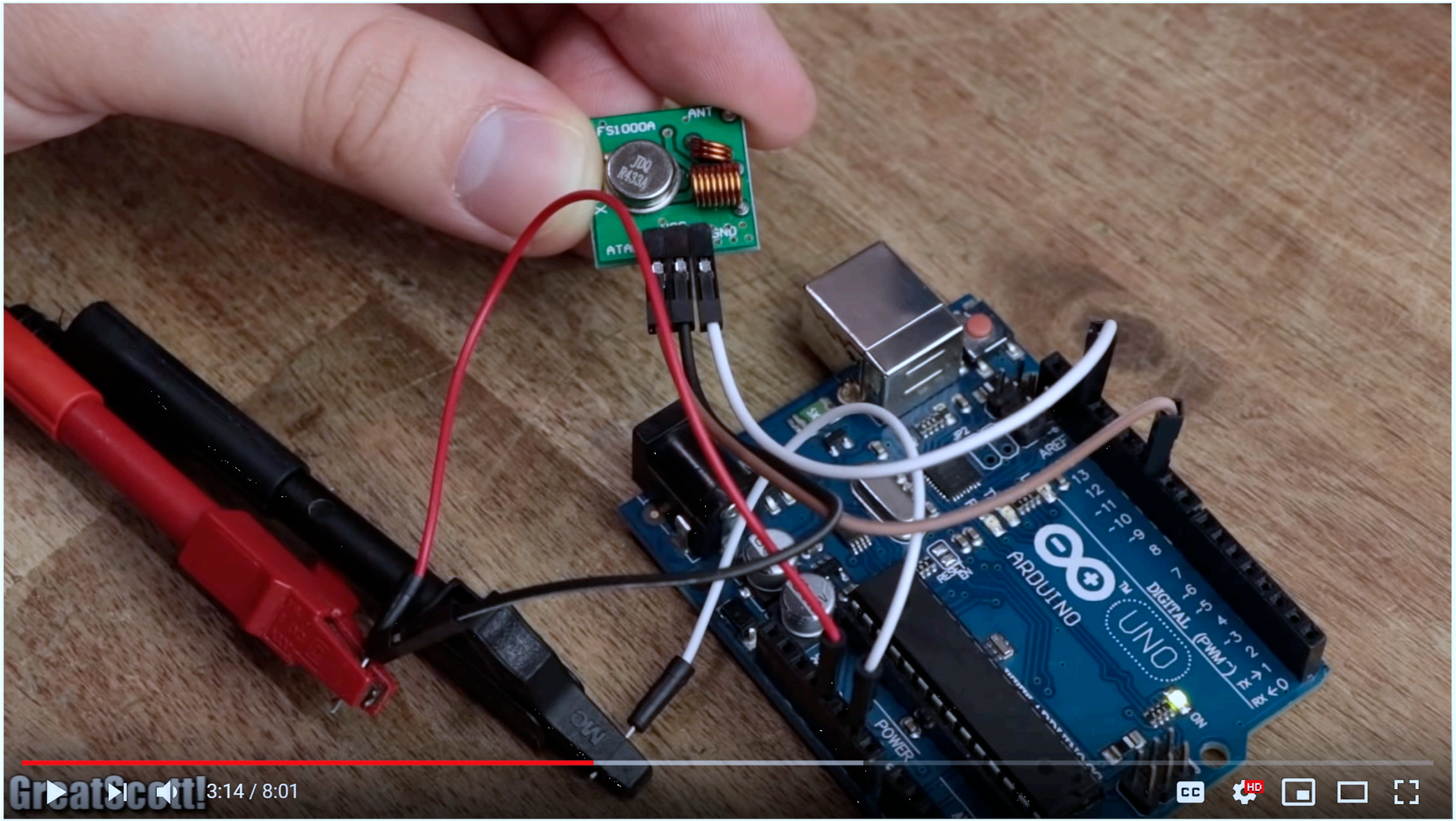

Digital Output → High/Low
Digital Input
PWM Output/ → High/Low
Analog Output → High - Low
Analog Input → High - Low
main functions
2:50 / 8:01
GreatScott!

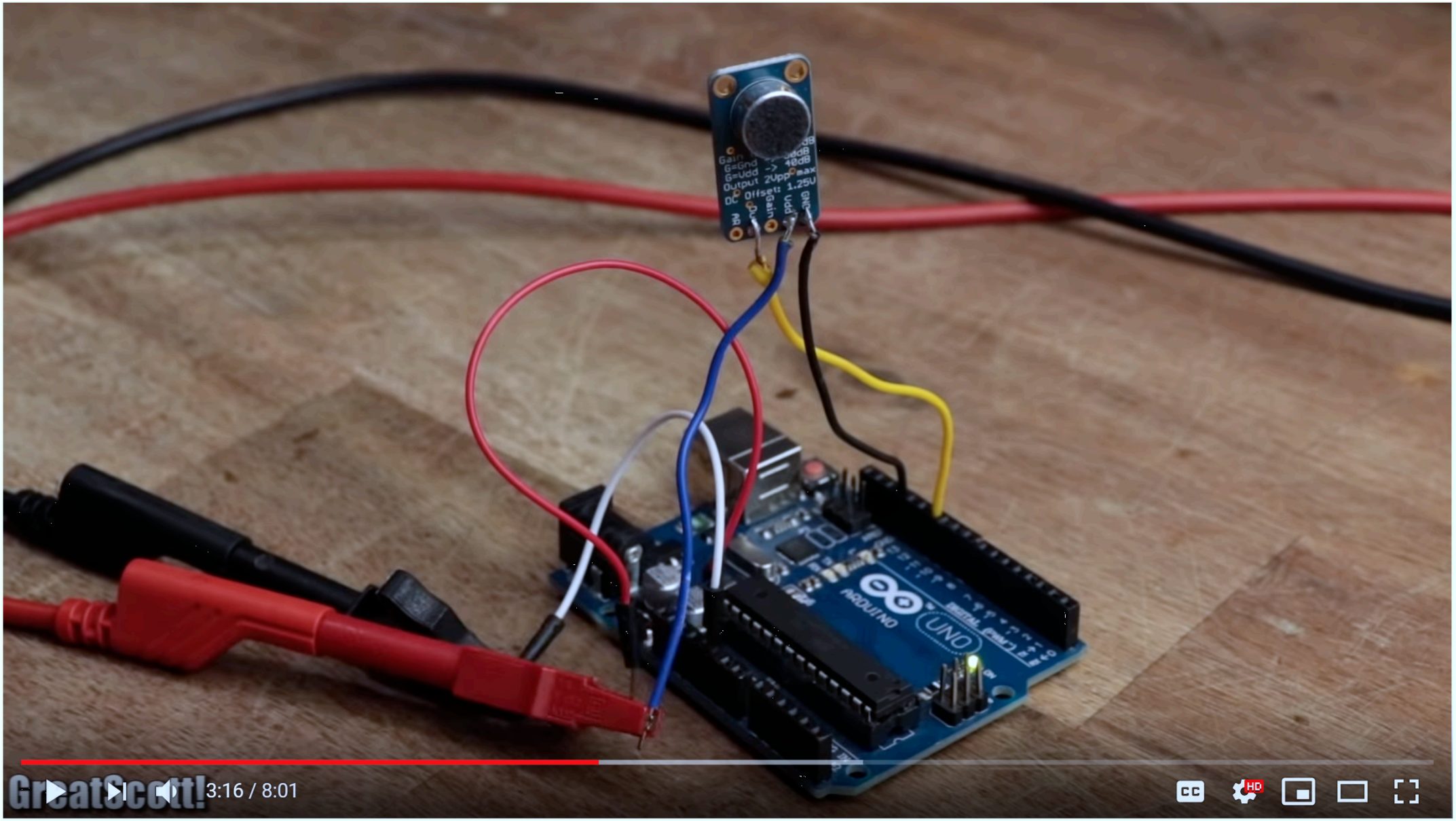

JP2
AREF
GND
13
12
~11
~10
~9
TX
RX
GreatScott!
2:56 / 8:01

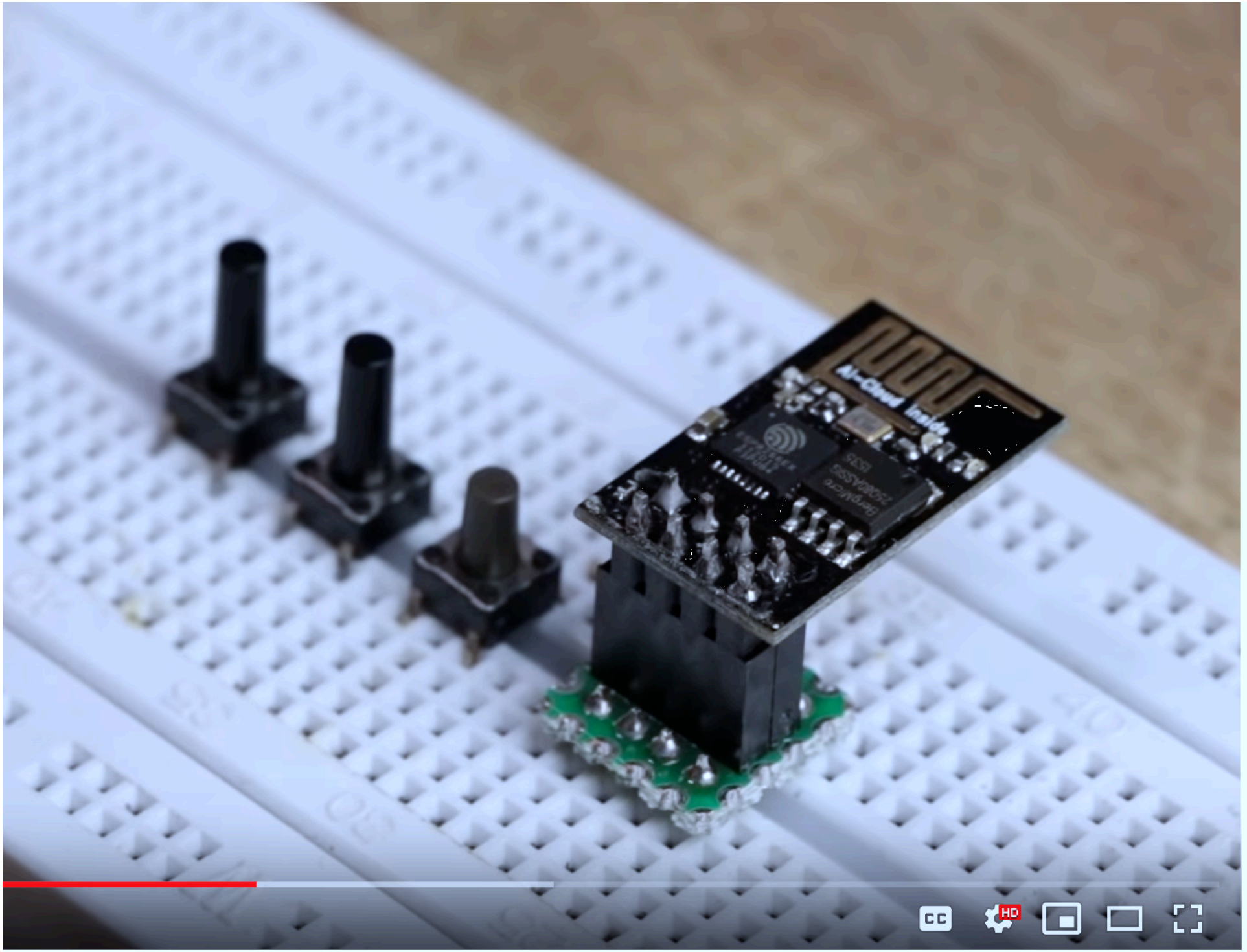

CS
47
25V
CS
47
25V
TX
RX
RESET
2:58 / 8:01

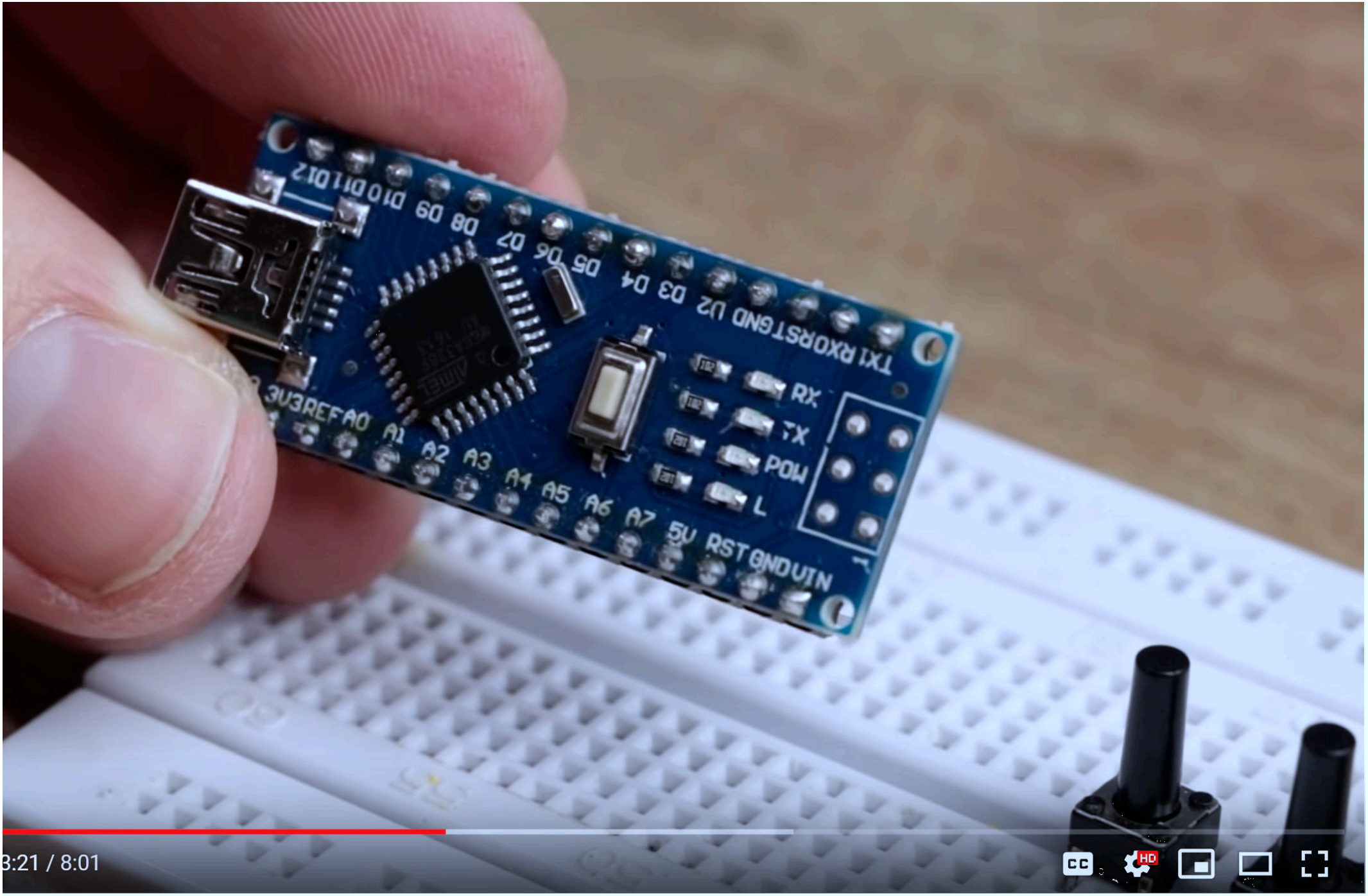

CS
47
25V
CS
47
25V
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
RESET
RX
TX
WWW.ARDUINO.CC
ARDUINO
3:04 / 8:01

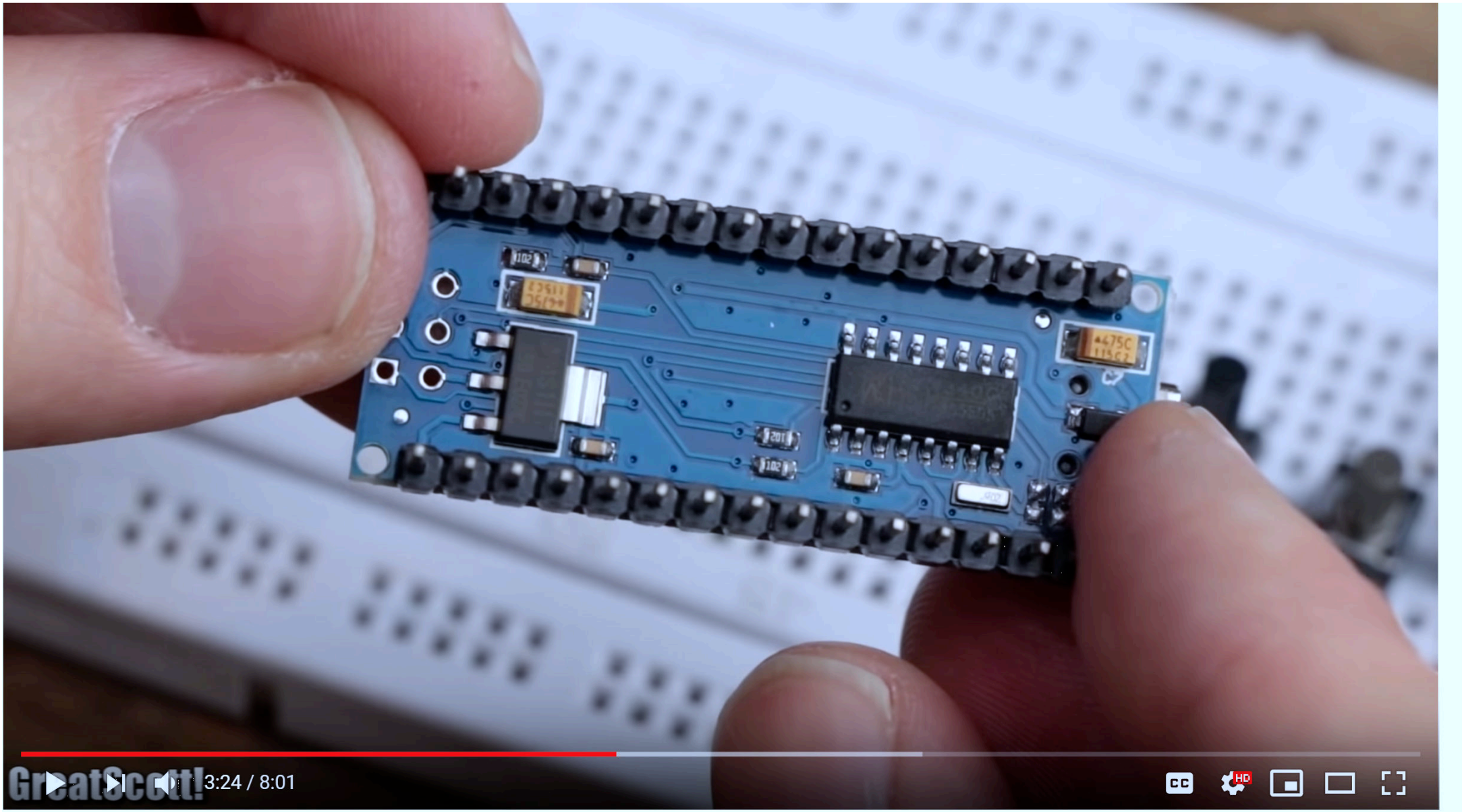

ELV
DPS 5315
Master
5.00 V
Active
0.044 A
0.22 W
U-Limit
5.00 V
I-Limit
1.000 A
Master
Slave
Dual
Series
U/I
GreatScott!
3:07 / 8:01

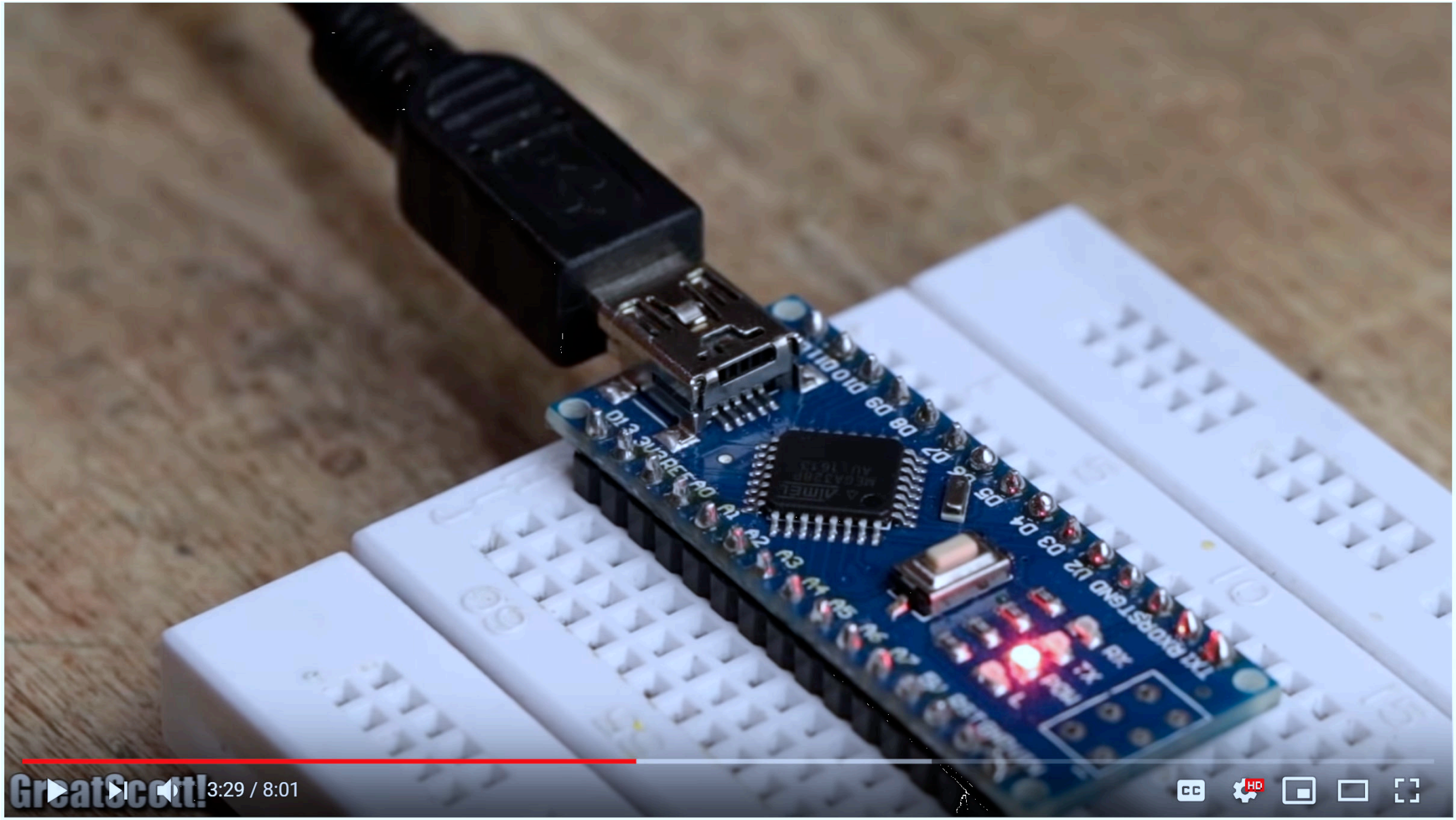

FS1000A
ANT
JDQ
R4S3A
ARDUINO
UNO
DIGITAL (PWM~)
POWER
ON
GreatScott!
3:14 / 8:01

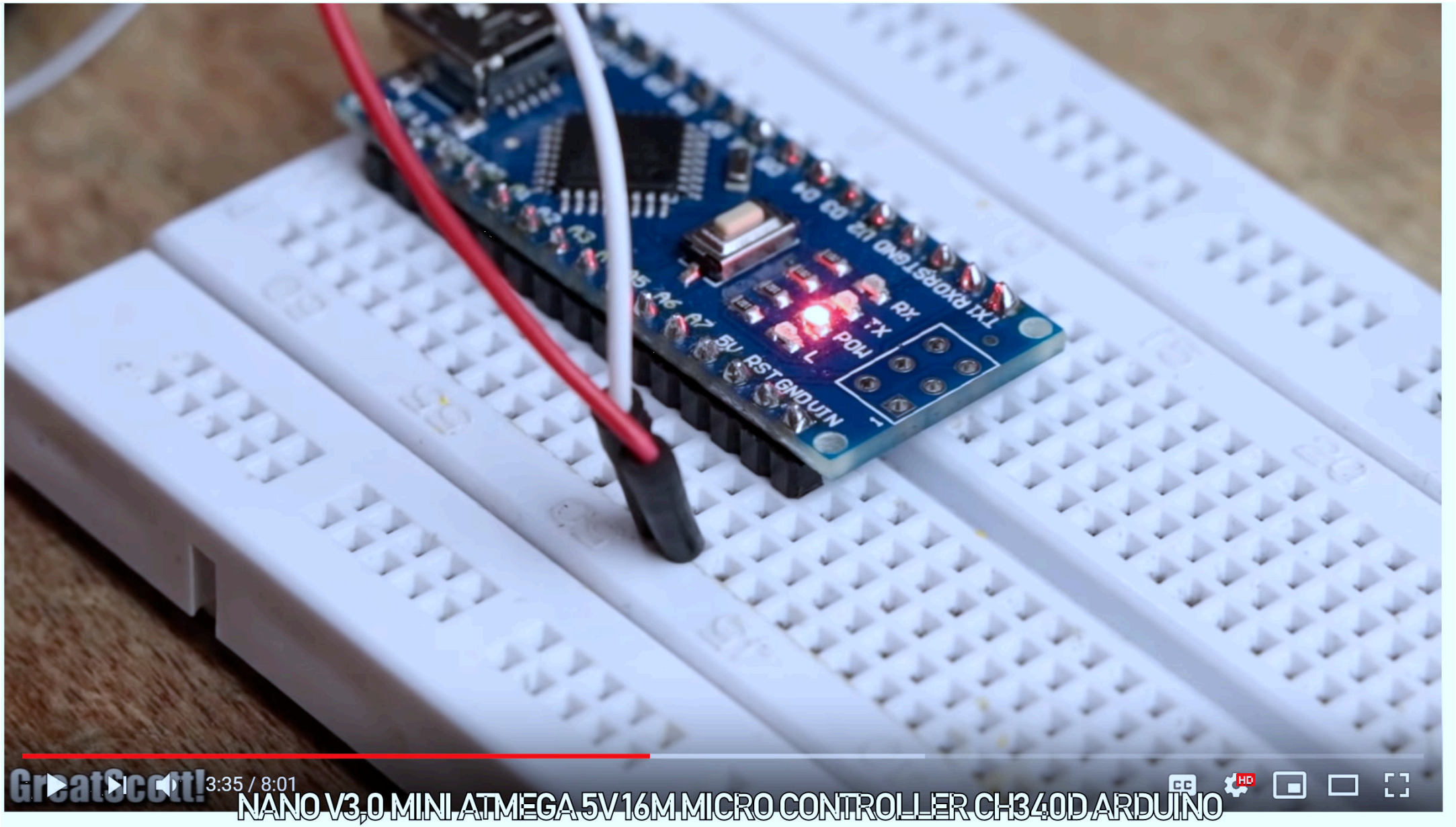

Gain
G=Gnd
G=Vdd
Output
DC Offset: 1.25V
Vdd
Gnd
ARDUINO
UNO
GreatScott!
3:16 / 8:01


Ai-Cloud Inside


TX1 RXO RST GND D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
RX
TX
POW
L
3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
3:21 / 8:01


GreatScott!
3:24 / 8:01


GreatScott!
3:29 / 8:01


GreatScott!
3:35 / 8:01
NANO V3,0 MINI ATMEGA 5V16M MICRO CONTROLLER CH340D ARDUINO

New UNO R3 Development Board ATmega328P for Arduino Compatible (with USB

★★★★★ 2 product ratings

Item condition: **New**

Quantity: 1 More than 10 available / 1,354 sold

Price: **US $6.89**

Buy It Now

Add to cart

189 watching

Add to watch list

Add to collection

1,354 sold	Experienced seller	More than 93% sold

Shipping: FREE Economy International Shipping | See details

See details about international shipping here.

Item location: Shenzhen, China

Ships to: Worldwide See exclusions

Delivery: Estimated between Tue. Mar. 14 and Wed. Apr. 12

Seller ships within 1 day after receiving cleared payment .

Please note the delivery estimate is greater than 15 business days.

Payments: PayPal VISA

Credit Cards processed by PayPal

PayPal CREDIT

Get more time to pay. Apply Now | See Terms

See details

Returns: 30 days money back or item exchange, buyer pays ret... shipping | See details

Seller

sk2store

99.4% P...

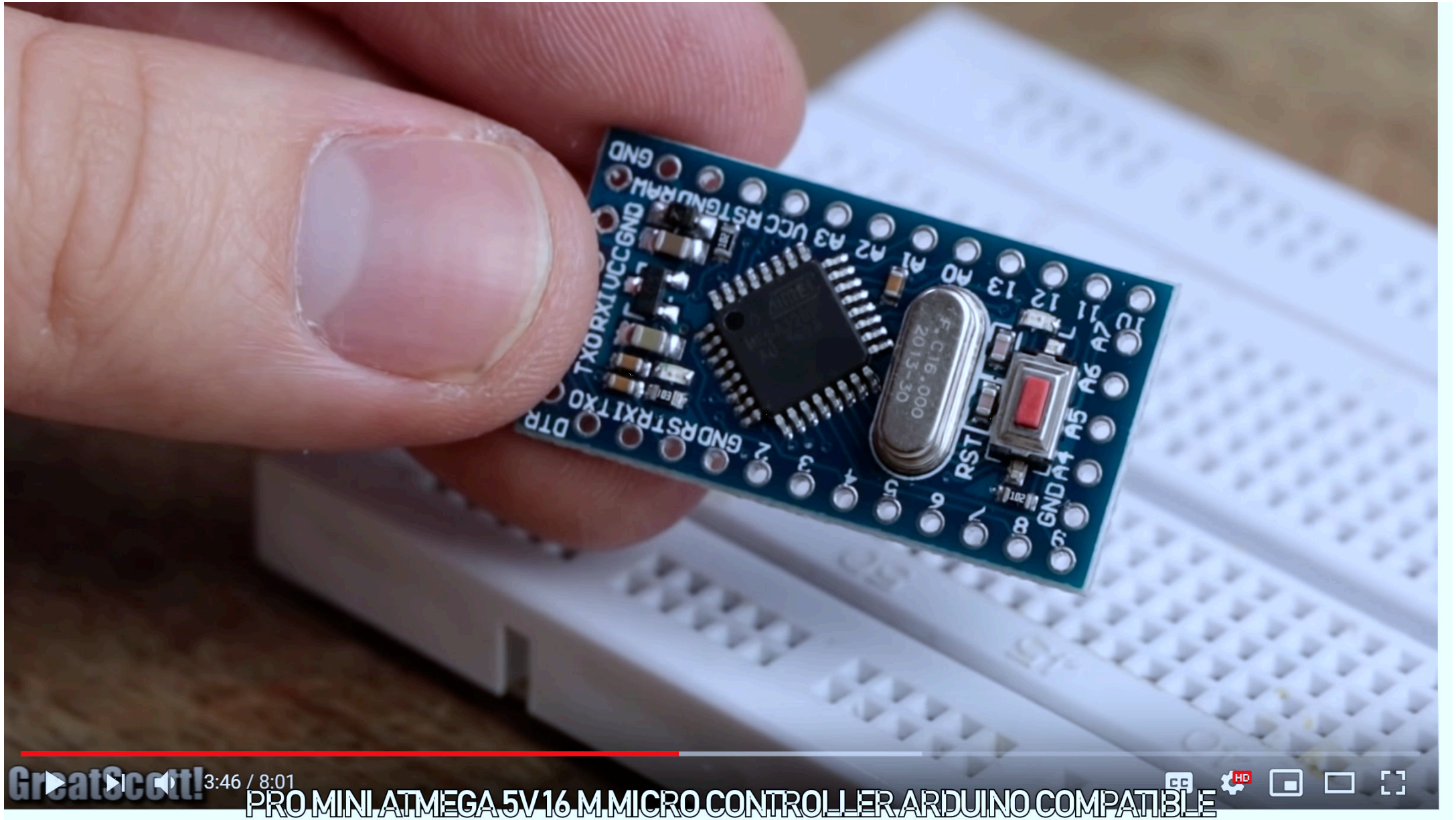
Follow

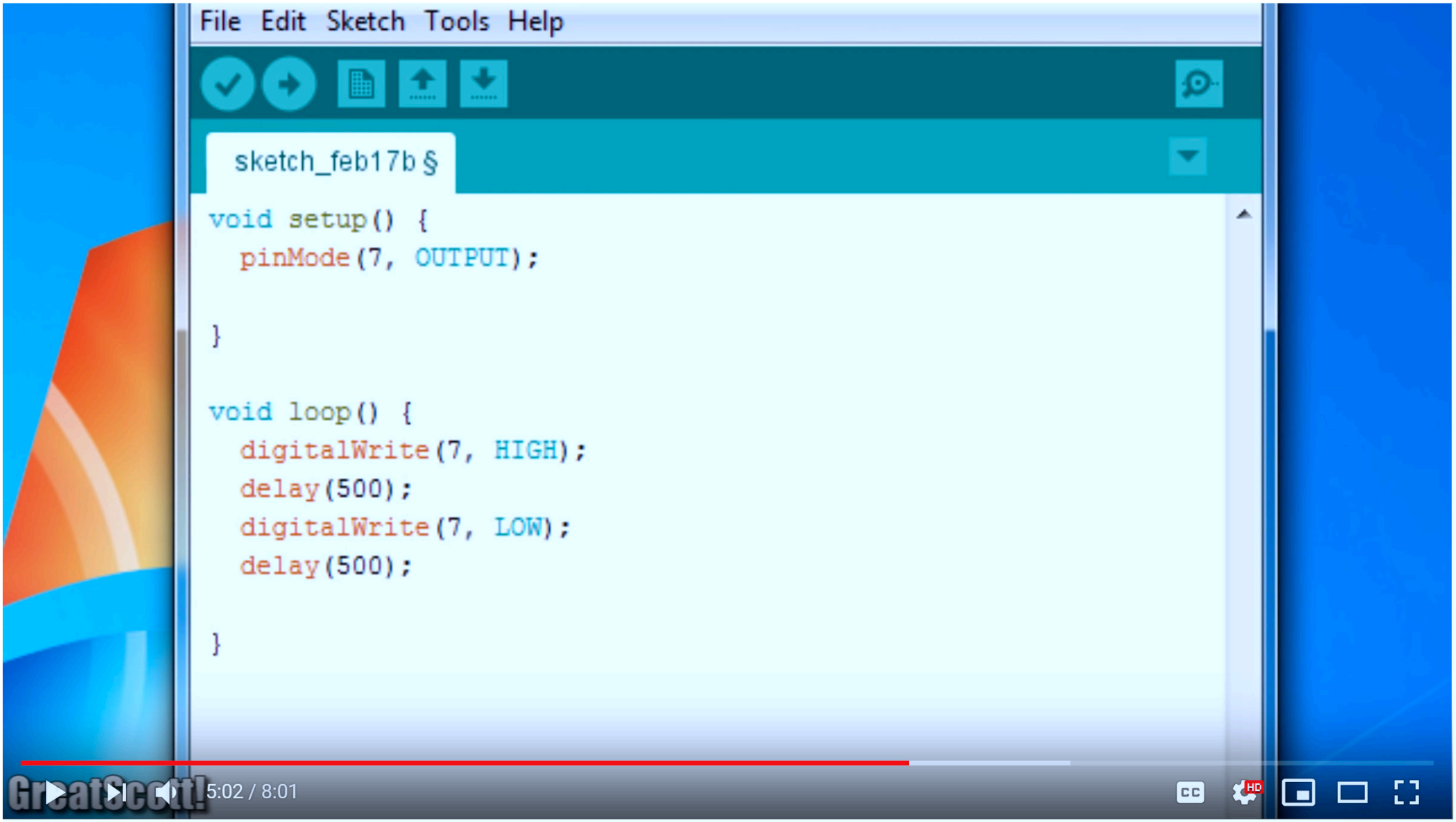
Visit sto...

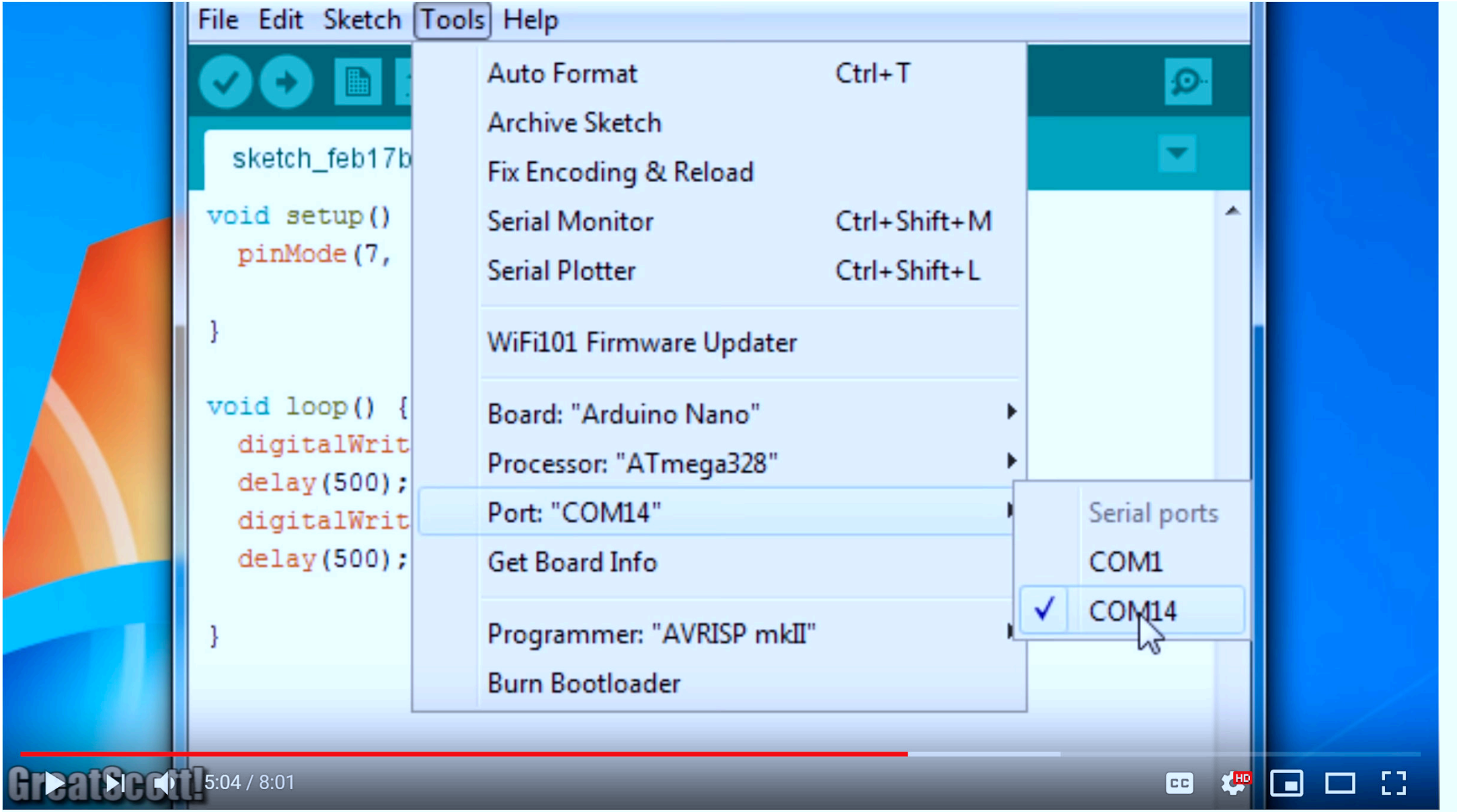
See oth...

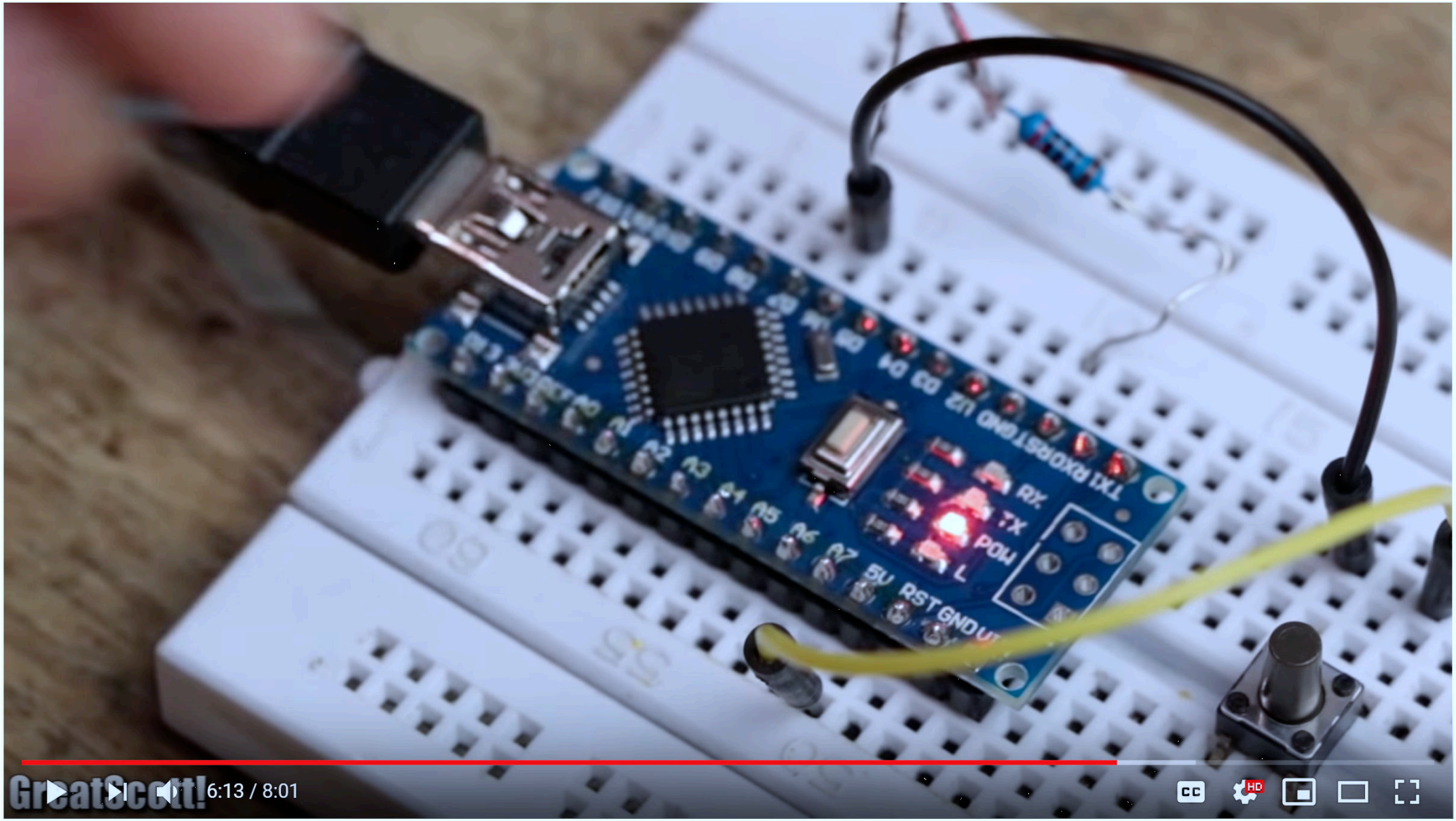
Have one to sell? Sell now

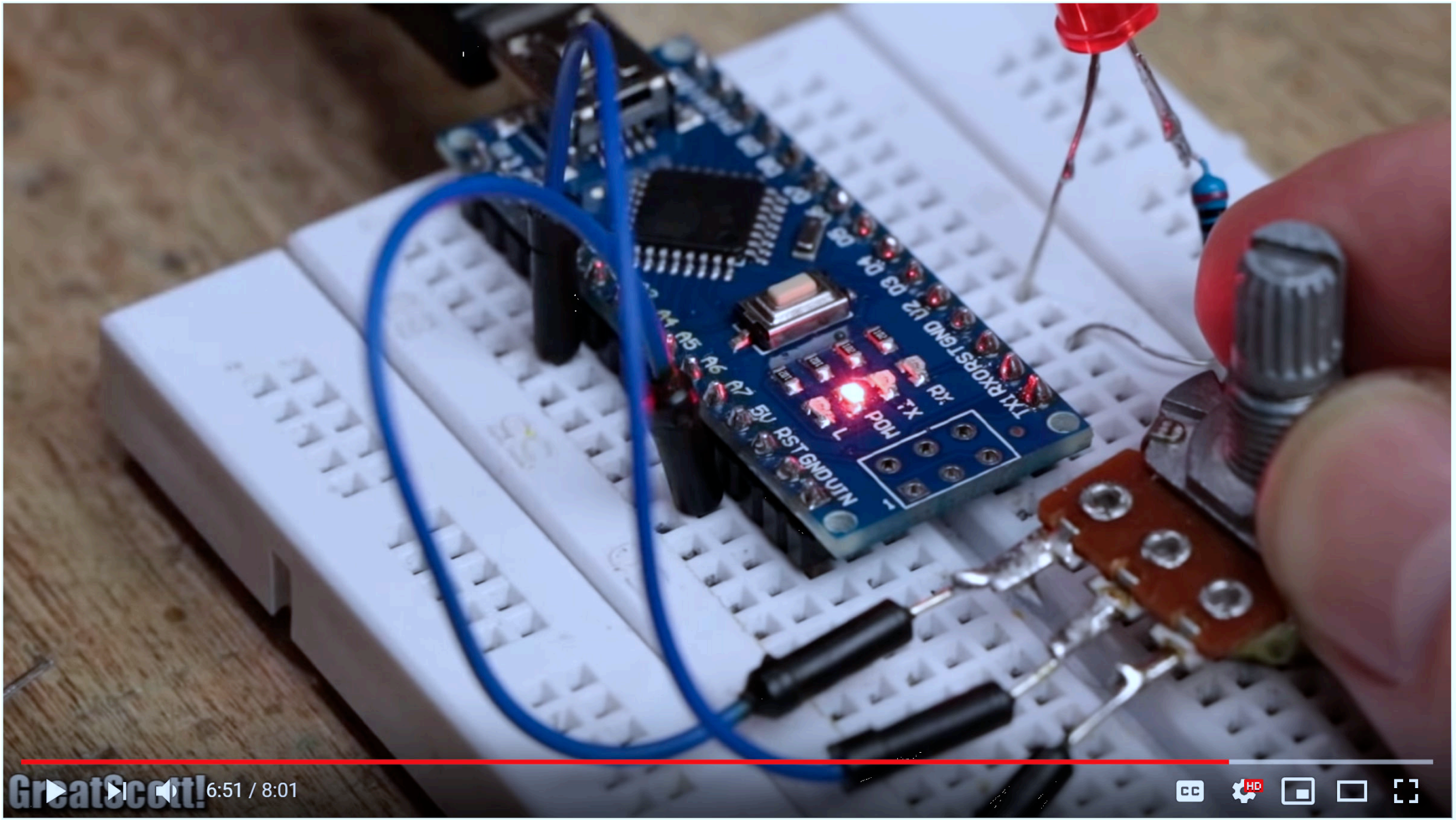
3:39 / 8:01


GND
RAW
RST
VCC
A3
A2
A1
A0
13
12
11
A7
A6
A5
A4
TX0
RX1
RST
GND
2
3
4
5
6
7
8
9
10
DTR
TXO
RXI
VCC
GND
F_C16.000
2013-30
RST
GND
GreatScott!
3:46 / 8:01
PRO MINI ATMEGA 5V16 M MICRO CONTROLLER ARDUINO COMPATIBLE


File Edit Sketch Tools Help
sketch_feb17b §
void setup() {
pinMode(7, OUTPUT);
}
void loop() {
digitalWrite(7, HIGH);
delay(500);
digitalWrite(7, LOW);
delay(500);
}
GreatScott!
5:02 / 8:01
HD


File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino Nano"
Processor: "ATmega328"
Port: "COM14"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM14
sketch_feb17b
void setup()
pinMode(7,
}
void loop() {
digitalWrit
delay(500);
digitalWrit
delay(500);
}
5:04 / 8:01


RX
TX
POW
L
GreatScott!
6:13 / 8:01


TXI RXO RST GND U2 D3 D4
RX
TX
POW
L
RST GND UIN
A4 A5 A6 A7 5V
GreatScott!
6:51 / 8:01


METRAHIT
TRMS PROFESSIONAL
PRO
MULTIMETER
02.907
DC
V
ZERO
ESC
MEASURE
SETUP
MAN / AUTO
DATA / MIN / MAX
ON / OFF
LIGHT
GreatScott!
6:55 / 8:01